PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 :
G06F 9/455

A1

(11) International Publication Number: WO 95/28673

(43) International Publication Date: 26 October 1995 (26.10.95)

(21) International Application Number: PCT/US95/05009

(22) International Filing Date: 18 April 1995 (18.04.95)

(30) Priority Data:
08/229,935 19 April 1994 (19.04.94) US

(71) Applicant: ORCHID SYSTEMS, INC. [US/US]; 103 Old Colony Road, Wellesley, MA 02181 (US).

(72) Inventors: HICKEY, Neil; 33 Canterbury Circle, Kennebunk, ME 04043 (US). ANTHONY, Robert, W.; 103 Old Colony Road, Wellesley, MA 02181 (US). SPILLER, Seth, A.; 62 Organug Road, York, ME 03909 (US).

(74) Agent: PASTERNACK, Sam; Choate, Hall & Stewart, 53 State Street, Exchange Place, Boston, MA 02109-2891 (US).

(81) Designated States: AM, AT, AU, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, JP, KE, KG, KP, KR, KZ, LK, LR, LT, LU, LV, MD, MG, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SI, SK, TJ, TT, UA, UZ, VN, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: TRAINABLE USER INTERFACE TRANSLATOR

(57) Abstract

An apparatus and method for converting a first user interface used for existing applications running on a host computer to a second user interface for use on a client computer. The apparatus intercepts prompts and request for input from the host, converts them to a form appropriate for use on the client computer, and passes the converted prompts and requests to the client. The apparatus can store information for use at a later prompt or request, branch on the stored value to vary path execution, and handle errors generated by incorrect input.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AT	Austria	**GB**	United Kingdom	**MR**	Mauritania
AU	Australia	**GE**	Georgia	**MW**	Malawi
BB	Barbados	**GN**	Guinea	**NE**	Niger
BE	Belgium	**GR**	Greece	**NL**	Netherlands
BF	Burkina Faso	**HU**	Hungary	**NO**	Norway
BG	Bulgaria	**IE**	Ireland	**NZ**	New Zealand
BJ	Benin	**IT**	Italy	**PL**	Poland
BR	Brazil	**JP**	Japan	**PT**	Portugal
BY	Belarus	**KE**	Kenya	**RO**	Romania
CA	Canada	**KG**	Kyrgystan	**RU**	Russian Federation
CF	Central African Republic	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CG	Congo			**SE**	Sweden
CH	Switzerland	**KR**	Republic of Korea	**SI**	Slovenia
CI	Côte d'Ivoire	**KZ**	Kazakhstan	**SK**	Slovakia
CM	Cameroon	**LI**	Liechtenstein	**SN**	Senegal
CN	China	**LK**	Sri Lanka	**TD**	Chad
CS	Czechoslovakia	**LU**	Luxembourg	**TG**	Togo
CZ	Czech Republic	**LV**	Latvia	**TJ**	Tajikistan
DE	Germany	**MC**	Monaco	**TT**	Trinidad and Tobago
DK	Denmark	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**US**	United States of America
FI	Finland	**ML**	Mali	**UZ**	Uzbekistan
FR	France	**MN**	Mongolia	**VN**	Viet Nam
GA	Gabon				

Trainable User Interface Translator

Background

This invention relates to integrating and translating software application user interfaces from a targeted computer system to a new computer system without modifying the underlying application.

Software developers, system integrators, value added resellers and end users are eager to utilize the leading edge handheld computers and portable data collection terminals. Yet, incorporating these new devices into existing software systems has proven difficult primarily because the computer system running the application must provide a user interface with a minimum functionality. For instance, an inventory system's user interface may require a screen capable of displaying 24 lines by 80 characters. Yet, a portable terminal may only provide 4 lines by 40 characters, and therefore be incapable of directly running the application. Since handheld or portable terminals typically do not provide the required minimum functionality, they are not capable of running most current applications.

The traditional solutions to this problem included rewriting the old application, buying a new application suited to the portable terminal restrictions, or writing a custom mapping program that would "map" the fields, prompts and responses from their respective positions on the targeted computer display to the more usable positions on the portable device. Rewriting the old application takes time, costs money and risks the introduction of bugs into the existing system.

Buying a new application involves significant expense and risk.

Custom mapping programs are expensive and time consuming to create and increase the maintenance cost of the application as changes in the application program could require further changes in the custom mapping program. In addition, custom mapping programs can create synchronization problems. The need to synchronize becomes obvious when you think about the impact of an error message not being seen by an operator who continues to type ahead. Therefore, custom mapping programs are not a satisfactory solution to the problem.

The present invention solves this problem by acting as an intelligent trainable interface between an existing application and a new computer system. The result is that the present invention allows the use of existing computer software with hardware for which it was not originally designed. Specifically, it provides a means of interfacing with the existing program, processing the data from the display screens of that existing program and presenting these data to the user in a different manner and/or format. Similarly, it accepts data from the user, reformats the data if necessary, and presents the re-formatted data to the existing application.

One goal of the present invention is to provide a system that can translate or convert an existing software application's user interface, so as to operate on a new computer system. In addition, it is a goal of the present invention to provide a system that utilizes a simple scheme to educate or train the system to translate an existing software

application's user interface. A further goal of the invention is to provide synchronization mechanisms to sync operation of a portable device to that of an existing software application. An additional goal of the invention is to provide robust error handling of application errors.

Summary of the Invention

By means of this invention, existing application software may be utilized on a system which does not provide the required level of user interface functionality.

The invention discloses an trainable apparatus for translating an existing software application's user interface. The apparatus comprises a computer adapted to communicate with both a host computer and client computer. The apparatus intercepts the host computer's input/output stream and translates the application user interface into a client user interface for use on the client computer. The computer is additionally adapted to simplify user interactions with the application by hiding repetitive tasks and redundant information conveyed by the application user interface. The computer is further adapted to unify host applications into a single user interface.

A method of creating and training the apparatus is also disclosed. The apparatus is trained by the monitoring of a user's interaction's with the application. The monitoring process creates a path history utilized by the apparatus for translating the application user interface.

Brief D scription of the Drawings

Fig. 1 is a block diagram of a prior art computer system utilizing handheld or portable terminals.

Fig. 2 is a diagram illustrating the functional difference between a portable terminal and a terminal targeted by the application.

Fig. 3 is a block diagram of a computer system utilizing the invention to interface to portable terminals.

Fig. 4 is a block diagram of the Virtual User paths created during the education process.

Figs. A-KK are screen printouts of one embodiment of the present invention.

Detailed Description of the Preferred Embodiment

Fig. 1 shows a prior art computer system using a portable terminal. A computer system 10 runs an application 12 specifically designed for a portable terminals 16. This application 12 interacts with the user 14 through the portable terminal 16. The application 12 communicates with the portable terminal 16 through a communications medium 18 such as infrared, radio frequency or even direct wire. The portable terminal 16 displays to the user 14 prompts requesting specific information. The user 14 enters replies into the portable terminal 16 in response to these requests.

If the application program is not designed to run on the portable terminal, the user will be unable to interact with the application. For instance, referring to Fig. 2, the application may require a terminal 20 with a full size keyboard 21 including functions keys, which keyboard is

different from the keyboard 23 available on the portable terminal 22. The application may also require a terminal 24 whose screen size is larger than the screen 25 on the portable terminal 22. In either situation, the application will be unable to run on the portable terminal without some form of change to the system.

Fig. 3 shows a computer system utilizing the present invention. A computer system 30 runs an application 32 that is not designed for use with a portable terminal. The present invention 34 is interposed between the application 32 and the portable terminal 36. The application 32 communicates through communications medium 33 with the present invention 34 which in turn communicates with the portable terminals 36 through a communications medium 38. Again, the portable terminal 36 displays to the user 39 prompts requesting specific information, but these prompts are generated by the present invention 34 and not the underlying application 32. The user 39 enters replies into the portable terminal 36 in response to these prompts, but these replies are again captured by the invention 34 and not necessarily passed directly to the application 32.

The present invention may reside on the same computer system as the underlying application program or may reside in a separate computer system and communicate with the application program through a network. In either situation, the present invention intercepts the I/O stream and translates the user interface for use on the portable terminal. The present invention may also be used to simplify access to an

existing program or group of programs even though access is through the same computer system as that of the existing program(s).

In order to interface and translate between an existing application and a portable terminal, the present invention creates a Virtual User (VU). A VU appears to the existing application as a real user. The VU is trained to operate the existing application, just as a real user would, and to present data to the real user in a format compatible with a portable terminal.

The VU is created, prior to use in the final system, by monitoring the interactions of a human user while operating the application. In the background, the present invention is creating a "path file" that the VU will use to automatically navigate through the particular task in the application. The path file consists of a series of automatically generated "steps" created during the training process that the VU later executes in a linear or branched sequence so as to move around the application. Every cursor stop in an application is reviewed by the VU and at least one step, possibly a series of steps, is associated with the stop.

The present invention provides three features to facilitate the creation of the VU. First, the present invention utilizes a menu driven interface to facilitate the process of creating the VU, making the process interactive as opposed to "off line". The user is not required to program in the classical sense, but rather just to interact with the application.

Second, the VU can create and utilize functions not available on the underlying application. Because the present invention is simply following a pre-defined set of steps, any combination of keystrokes a person can enter, a VU can also. This allows the VU to be trained or educated to navigate around the entire application, enter and exit different applications, and even access and exit different hosts. For example, an application may have two menu options: the first menu would list the quantity of item on hand and second menu would list the location of item. A typical user directly accessing the application would have to enter the first menu option to get the quantity, exit the first menu option, enter the second menu option to get the location just to get both pieces of information. The VU can be trained to do the same steps but present the user with one screen that has both pieces of information. Therefore, the present invention can create new functions for a user that were not previously available by running a particular application "straight through as written".

Third, the present invention provides streams monitoring that allows the VU to synchronize itself with the application. The VU must be synchronized with every application cursor stop prior to sending any keystrokes back to the host. The use of streams monitoring allows the VU to "recognize" the current cursor position and hence determine on what screen of the application it is "looking at".

Creating and Educating the Virtual User

In order to translate an applications user interface, the VU must be created and educated.

The following is a list of all the commands accessible from FILE, VARS, HOST, CLIENT and MISC selections of the menu bar i\of the present invention during the education process.

FILE MENU

Save path File : save the steps that you have created to the ***path file***, without exiting

Save & Exit : save the path file and exit

Quit : exit without saving

Save Window : allows you to save the image of any screen for later printing

VARS MENU

Declare Variable : declare a variable to be used in the path file you will create

Set Variable : initialize a variable already declared to a starting value

Branch on Variable : define the ***path name*** the program will branch to when a variable equals the "to match" filed defined here. NOTE: "No." selects the *step* # in the ***path***.

HOST MENU (first time)

Connect Local (PTY) : defines connection method to host application as ***pseudo terminal*** access to the same computer the VU is running on.

Connect Local (Pipe) : defines connection method to host application is via a ***pipe*** to the same computer the VU is running on.

Connect via TELNET : defines connection method to host application is via ***telnet,*** in this case the application is running on different computer than the VU.

Connect via Serial : defines connection method to host application is via a ***serial port*** on the computer the VU is running.

HOST MENU

Send to Host : send any combinations of keystrokes to the host

Wait for Host : synchronization step that makes sure the VU and the host application are at the exact same character of a given application at a given time

Save Host Screen Data : save a particular ***window*** of the host screen, often used to store error messages that appear on the

same line of the screen every time.

Bypass Virtual User : allows you to stop the interactive training and key data directly into the host application.

CLIENT MENU

Clear Screen : send the command sequence to clear the screen

Sound Tone : send the command sequence to sound the bell tone

Move Cursor : send the command sequence to move the cursor to a specific x, y coordinate of the Client screen

Send Message : send a string of characters to the Client for display

Get Reply : request a reply from the Client (Scanner or keyboard input)

MISC MENU

Start New Path : defines the current step as the beginning of a path, used as the ***connection point*** when using the ***connect*** statement below.

End Path : defines the end to a path

Connect (Loop) : allows program flow to be redirected to a path label specified using the "Start New

Path" option above.

Exec Procedure : allows program control to pass to a pre-defined procedure, procedures are available for ***Terminal Mode,*** Logon, etc.

By selecting various options from the menu bar defined above, one can access the application for which the VU will be trained. After accessing the application, one can interactively train the VU to react appropriately to prompts in the application.

An important concept is ***application program flow.*** Application program flow is the sequence of data input prompts that must be followed for use of a particular application. These prompts include menu selections, data and time entries, and other inputs as is required in order to operate an application. The present invention provides a method of pre-defining and remembering how someone interacts (e.g. a Virtual User) with an application.

The first step for a person using any application is to be trained on how to interact with the application. The VU, however, can be trained to remember keystroke sequences that are repetitive, access system time and date information - never forgetting how to do it. The VU must be trained on how to react to ***cursor stops*** that an application makes. After being trained in how to react to the cursor stops/prompts, the VU will be able to operate the same functions in the application by itself.

To facilitate data input, the VU must be trained to prompt for Client data input. Clients are the terminals (RF handhelds, regular terminals, etc.) where an operator will enter data. The VU simplifies the operation of an application without requiring any programming changes in the application. The VU does this by automating the majority of the application cursor stops/prompts and only passing through to the Clients that information required for the particular task at hand. At each cursor stop in the application there is the option of prompting Clients for data or doing any of the other functions the menu bar allows. This option allows the VU to vary its response to a given application cursor stop. In this way, the present invention ensures that the can branch in the execution of steps.

The following is a list of the basic steps of a typical Virtual User training session.

1) **Host Connection/logon/application access**
2) **Client Menu**
3) **Host application access**
4) **Automated navigation through access menus**
5) **Prompting for Client data input**
6) **Branching and error instruction based on Client data**
7) **Looping to start the Client task over again or returning to a Client Menu**

By repeating these parts for whatever *function* you are trying to perform you can create simplified user input screens

for the portable terminal operators. Because the VU is simply navigating through an application the same way a person would only at 66MHz (the clock speed of the computer) you can train it to do tasks that you would not expect a person to be able to perform in a timely manner. This simple concept allows you to put together functionality from several menu options of an application (even several different applications) and create a single operator screen that might not even exist in the original application.

The details of each of these steps will become more clear through the following example training session. Note that the figures referred to in the example below show three important areas. Each figure represents the training screen of the present invention. At the top of the figures is the menu bar used to access the function menus describe above, as illustrated by Fig. A-a. In the middle of the figures is a model Client window labelled "CLIENT", reflecting the size and shape of the screen on the Client computer system, as illustrated by Fig. A-a. This screen is defined in the Spec file. An example Spec file is included as Appendix B. Finally, a "Host window shade", labelled "HOST", will pop up as needed to show messages and prompts sent by the host. This is illustrated by Fig. I-c. The CLIENT, HOST and invention menus show the state of the system during the education process. When the education process is finished, the completed VU will operate as trained.

Example Virtual User Training/Education Session

The following is a simple application example as run by an actual user to illustrate the creation of a VU. Throughout the following text [] are used to represent selections that are to be selected using the pull down menus or to represent actual keys to press. For example: select [Start New Path] means to highlight the selection "Start New Path" in a pull down menu, press [end] means to press the end key.

Step 0: (Fig. A-a through A-d)

Access the **Misc.** menu and select the [Start New Path]. You must define the name of the *path* you are starting, this is important for looping and branching control. The path name specifies the point in the path for branch and loop access. In this example the name [main_menu] has been selected since this is the point at which the main menu will be displayed.

Step 1: (Fig. B-a through B-b)

Access the **Client** menu and select [Move Cursor]. This will set the cursor position in the client window. The default settings of row: 1 and col: 1 are have been selected. The cursor in the client window moves to the x,y position or 1,1 as a result.

Step 2: (Fig. C-a through C-b)

Access the **Client** menu and select [Clear Screen]. The client screen is cleared as a result.

Step 3: (Fig. D-a through D-h)

Access the **Client** menu and select [Send Message]. A submenu is displayed. Access the **Type** submenu and select [Text] and enter the message to be displayed on the client screen. In this example the message [Do you want to run the program y/n] is entered. The text "Do you want to run the program y/n" is now displayed on the client screen, split into lines of length appropriate to fit on the Client screen. The system is now ready to get a reply from the client.

Step 4: (Fig. E-a through E-c)

Access the **Vars** menu and select [Declare Variable]. A variable may be declared to store the response from the client to the question in Step 3. The variable name [answer] is entered.

Step 5: (Fig. F-a through F-b)

Access the **Client** menu and select [Get Reply]. Type [answer][tab][1] to enter "answer" as the variable the response from client will be stored into and sets the maximum length equal to one character. Note that the [backspace] now works as would normally be expected for editing mistakes.

Step 6: (Fig. G)

There are two options that need to be defined. The first is for a "y" or yes response, in general a typical training procedure would be to go through the "normal" path required to operate the application. In this case normal means "y", so we

will respond with a "y", later in the example the training for the "n" can be entered.

Step 7: (Fig. H-a through H-c)

Access the **Vars** and select [Branch on Variable]. Type [answer] [tab] [y] [tab] [start_rw] [tab]. This defines the action: if the variable "answer" equals the value "y" then proceed with the path named "start_rw". In the next menu select [d] to declare a pathname. At this time the "n" action may also be defined. Type [n] [tab] [exit] [tab] [d] which causes the path "exit" to be run when "answer" is equal "n".

Now the path named "start_rw" is created.

Host Connection/logon/application access: for our training example we will be running both the application and the present invention on the same host. The application is therefore referred to as running on a ***local host*** as opposed to running on a different or ***remote host.*** The following instructions apply to accessing an application via a pseudo terminal (PTY) on a local host only. You have to be logged onto the local host in order to run the present invention, therefore, with this method of connection you are not required to logon to the host prior to accessing the application. Refer to the applicable section for other connection methods.

PATH NAME: start_rw

Step 0: (Fig. I-a through I-c)

Access the **Host** menu option. Once you are "in" the menu bar you may use the arrow keys to navigate. Use the down arrow to "pull down" the HOST ***window shade***. Select [Connect Local (PTY)] by highlighting this option and pressing [enter]. You will now be prompted for **Command:** enter the keys you would normally use to access your application, for real world type [go_rw], you can leave the **Argument:** line empty, press [end] to compete this step. You will notice that when you press [end] the Virtual User sends your keystrokes as a command line entry to the host. You should now see the first screen of the Real World application in the ***terminal*** (middle) ***window.***

Alternate using [Exec Procedure]

Step 0:

Access the **Misc** menu and select [Exec Procedure]. Type [go_rw] [end].

Automated Navigation through access menus: you are now ready to train the Virtual User how to navigate through your application. As you will see, the first step to perform at ***cursor stops*** is a "Wait for Host" step. This operation makes sure that the Virtual User and host application are synchronized with each other. If you try to send information to the host it will ask you to perform a "Wait for Host" first. The order that you prompt for Client input is entirely

up to you, this tutorial follows some basic guidelines.

PATH NAME: nav_menus

Step 0: (Fig. J-a through J-c)

Access the **Misc** menu and select [Start New Path] [enter]. Type [nav_menus] [end]. NOTE: "nav_menus" was chosen as a name because this path will navigate through the menus required to access our desired transaction. The underscore is required in the name, no spaces are allowed.

Step 1: (Fig. K-a through K-c)

Access the **Host** menu and select [Wait for Host]. Press [enter] and use the down arrow to select [automatic], press [enter] [enter]. You have now confirmed an automatic host synchronization for the string "o continue, or ESC to exi" to appear at the bottom of the Host screen.

Step 2: (Fig. L-a through L-c)

Access the **Host** menu and select [Send to Host]. In the submenu select [Special Char(s)]. Press [enter], you should see <cr> in the window, press [.] [enter], edit using the [tab] key where necessary, press [end] to activate this step. You should see the application respond to the VU's carriage return with the next screen, now you need to enter your initials.

Step 3: (Fig. M)

We could easily "hardcode" a set of initials by using the sequence in step #2 or we could prompt the Client for "User

Initials". To prompt the Client for user initials, proceed as follows. Access the **Client** menu, use the down arrow to select [Clear Screen]. The Client screen should now be blank, you might not see any change because the old prompt "Real World (y/n)?" is hidden behind the terminal window.

PATH NAME: item_transfer

Step 0: (Fig. N-a through N-c)

Access the **Misc** menu and select [Start New Path]. Type [item_transfer] as the name of the path. This path will prompt the Client with information to complete an item transfer transaction. The program will give you the option to link the last path "access" to the new path "item_transfer". Press [n] to select "link to New Path".

Step 1: (Fig. O)

As in prior steps, we will clear the screen. Access the **Client** menu and select [Clear Screen]. The Client screen should now be blank.

Step 2: (Fig. P)

As above, access the **Client** menu and select [Move Cursor]. Select row 1 and column 1. The cursor should now blink at the x,y position of 1,1 on the Client screen.

Step 3: (Fig. Q-a through Q-b)

Access the **Client** menu and select [Send Message]. Select

[Text] in the submenu. The message to be displayed on the Client screen is now entered. Enter [item:]. The prompt "item:" now appears on the Client screen.

Step 4: (Fig. R)

As before, access the **Vars** menu and select [Declare Variable]. The variable for storing the Client response to the item prompt is now declared. The name [item] is entered as the name of the variable.

Step 5: (Fig. S-a through S-b)

Access the **Client** menu and select [Get Reply]. The replay from the Client will now be entered and stored into the variable "item". Type [item] [tab] [10] to enter "item" as the variable for the Client's response and a maximum length of ten characters will be allowed. The menu bar will be replaced with **Waiting for Client Input at Client or here.** A valid item number may now be entered.

PATH NAME: update_host.

Step 0: (Fig. T)

Access the **Misc** menu and select [Start New Path]. Type [update_host] as the new path name for this sequence of steps. This path will update the host with the response received under "item_transfer" path

Step 1: (Fig. U)

As before, access the **Client** menu and select [Clear Screen] to clear the screen at this time.

Step 2: (Fig. V)

As before, access the **Client** menu and select [Move Cursor]. Select row 1 and column 1 as the new location. The cursor in the Client window should now be positioned at the x,y location of 1,1.

Step 3: (Fig. W)

As before, access the **Client** menu and select [Send Message]. In the submenu select [Text]. Now the message to be displayed on the Client screen is entered. Enter [Processing Data]. The prompt "Processing Data" is now displayed on the Client screen.

Step 4: (Fig. X-a through X-b)

Access the Host menu and select [Wait for Host]. Select [automatic]. This will now automatically synchronize the Host and Client. Note that error handling will be provided in the "None of the above" path of "item_error" and that the time out is set for 100 x .1 sec = 10 seconds. Therefore, on an error condition, if 10 seconds elapses without a response, then item_error will be executed.

Step 5: (Fig. Y-a through Y-b)

Access the **Host** menu and select [Send to Host]. The

variable "item" will now be sent to the host. Select [Variable] from the submenu and type [item] as the variable to send to the host. Select [Special Char(s)] from the submenu and enter a [cr]. This will pass a carriage return to the Host.

PATH NAME: item_error

Step 0: (Fig. Z-a through Z-b)

Because data was entered that caused the application to go into its error handling routine, the cursor is no longer at its normal "next entry" position beside the second prompt "warehouse". Instead it is sitting at the bottom right hand corner of the screen with a message "Press ENTER or F8". Note that the error message "Item not on file" is displayed at the left hand corner of the screen. Step 4 of "update_host" had an error handling routine named "item_error" defined as the path to use if "none of the above" condition is true. Therefore, control has been passed to "item_error" to handle the error condition. Access Host and select [Wait for Host]. You have now confirmed an automatic host synchronization.

Step 1: (Fig. AA)

As before, access the **Vars** menu and select [Declare Variable]. This variable will store the host data that is currently in the "error window" on the host screen. Type [item_error] to declare the variable "item_error" which will be used in the next step.

Step 2: (Fig. BB)

Select the Host window shade and select "save host data". Enter the variable declared in step 1 and use the arrow keys to move the "window" over the error message on the screen. Use the shift "+" or "-" keys to increase or decrease the size of the "window" defined by the square brackets in the "Save Host Screen Data" box.

Step 3: (Fig. CC)

As before, access the **Client** menu and select [Clear Screen]. The Client screen should now be cleared.

Step 4: (Fig. DD)

As before, access the **Client** menu and select [Move Cursor]. Set the row to 1 and the column to 1. The cursor should move to the x,y location of 1,1 in the Client screen.

Step 5: (Fig. EE)

Select [Sound Tone] from the **Client** menu. A tone will be issued to the Client to notify the user of an error.

Step 6: (Fig. FF)

As before, access the **Client** menu and select [Send Message]. In the submenu select [Variable]. Enter "item_error" as the variable name and select [Special Char(s)] and define a carriage return and line feed so the next line of text does not overwrite the error message. Select [Text] and enter the message you want displayed on the Client screen.

Select [Special Char(s)] and enter a carriage return, line feed and the last part of the message to be displayed. In this example, the text "<cr> <lf> Press ENTER to <cr> <lf> continue:".

Step 7: (Fig. GG)

As shown in the above steps, declare a new variable "answer" to get the ENTER key from the client in order to continue.

Step 8: (Fig. HH)

As demonstrated in the previous steps, get the Client reply.

Step 9: (Fig. II)

Wait for the Client response to the prompt.

Step 10: (Fig. JJ)

Wait for the Host prior to sending the "answer" just entered from the Client.

Step 11: (Fig. KK-a through KK-b

The Host is now back at the item input prompt. Access the **Misc** menu and select [Loop] and connect this error path to the beginning of item_transfer to once again prompt the client for an item number.

Note that any error condition can be handled with this technique. It is equally correct to use any of the "Possible Response" areas of the Wait for Host dialogue box. Therefore, 5 additional "known" responses can be declared and "trained" in a similar manner as above. If you only want to train the VU for one error response that all conditions will use, then the "none of the above" option is appropriate.

Fig. 4 illustrates the interconnection of the various paths described in the above example. The VU trained as described above is now ready for use as an interface between the example application and a portable terminal.

Path File Primitives

The Path file specifies the sequence of steps and commands captured during the education process of the VU. Appendix A contains the Path file for the example VU illustrated above. The following primitives are used to record the host program behavior and the interaction with the human operator:

$endact:	terminate path file
$cl_connect:	establish client connection
$cl_clear:	clear client screen
$cl_tone:	sound bell tone on client
$cl_move:	position client cursor
$cl_send:	send data to client
$cl_get:	get data from client
$cl_discon:	break connection with client

$host_send:	send data to host application
$host_connect:	establish connection to host application
$host_discon:	break connection with host application
$host_save:	store data from emulation area
$host_sync:	monitor host operation and compare against behavior database
$declare:	declare a storage variable
$param:	modify as system parameter
$set:	change the value of a variable
$pipe:	establish transparent connection between client and host application
$new_path:	begin a new sequence of operations
$loop:	transfer control
$done:	terminate a path
$vbranch:	conditionally branch on variable value
$exec:	execute a program procedure

Spec File Primitives

The Spec file captures the system dependant information for use by the VU. Appendix B contains the Spec file for the example VU illustrated above. The following primitives are supported in the "program level" interface:

nop:	no operation
mark:	label program location
label:	label program location
begin:	start a program structure
end:	end a program structure

write:	output data to host or client
display:	output data to client
send:	output data to host
file_write:	output data to disk file
get:	input data from client
read:	input data
find:	locate data on emulator screen
if:	conditional execution
else:	conditional execution
endif:	conditional execution
declare:	declare storage variable
param:	set system configuration
set:	set variable's value
eof:	terminate spec file
connect:	establish connection to client or host
disconnect:	break connection
monitor:	monitor data from host or client and compare against expected data
pipe:	transfer data between host or client and monitor
configure:	set emulation, client I/O, or host I/O configuration
sleep:	delay
capture:	enter training mode
include:	insert commands from sub-file
declare_action:	process and store a pathfile
run_action:	execute pathfile
file_open:	open a disk file

file_close: close a disk file

spawn: execute a system command as a subprocess

goto: flow control

loop: flow control

break: flow control

exit: flow control

gosub: flow control

return: flow control

next: flow control

then: flow control

As disclosed by the example above, the operation of the existing program(s) is performed by the Virtual User program rather than an actual user. Thus the following functions are supported:

1. The sequence and format of the data obtained from the actual user may differ from that required by the original program.

2. User data may be combined with data from other sources for presentation to the existing application.

3. Data may be combined from multiple functions of a given application or multiple applications on a single host or even from multiple host computers for presentation to the user. Data may also be passed from one host application to another application with or without user interaction. This allows the generation of new, more complex functionality without writing new application programs.

4. The virtual user is trained with an interactive real-time, menu driven, manner. The present invention

monitors the actual behavior of the target host application and stores the details of this behavior for future reference. Many aspects of host program behavior are recorded including key elements of the actual data stream as well as the contents of the emulated screen displays. This behavioral database is used by the VU module to successfully operate the host program (including the detection and interpretation of error conditions) as well as being used as a source of data for presentation to the human operator and/or passed to different data entry screen within the application and/or different host applications.

5. The VU module is able to detect unexpected actions by the host program and report them as exception conditions to the operator and/or learn the behavior and the required response.

The present invention has been implemented in the ANSI C programming language. The present invention runs under the SCO Unix operating system (Santa Cruz, CA) running on a 486 class workstation. The client portable terminal is any device capable of running an ANSI or VT100 terminal emulator. Such a device is the Janus J2010 Handheld (Everett, WA).

It will be appreciated by those of ordinary skill in the art that the present invention can be embodied in other specific forms without departing from the spirit or essential characteristics thereof. The presently disclosed embodiment is therefore considered in all respects to be illustrative and not restrictive.

```
// Example Path File
//
DECVAR answer
DECVAR item_error
DECVAR item

NEW_PATH "main menu"
ACTION $new_path "main_menu" 0

ACTION $cl_move "main_menu" 1
    1 1
ACTION $cl_clear "main_menu" 2
    Y
ACTION $cl_send "main_menu" 3
    N "Do you want to run" ;
    <cr> <lf> ;
    "the program (y/n):"
ACTION $cl_get "main_menu" 4
    answer N 1
ACTION $vbranch "main_menu" 5
    answer 5
Y 0 "start_rw" Y "y"
Y 0 "exit" Y "n"
N N
N N
N N
EDISPATCH

NEW_PATH "start_rw"
ACTION $new_path "start_rw" 0

ACTION $h_pty "start_rw" 1
    "go_rw" ; ""
ACTION $loop "start_rw" 2
    Y 0 "nav_menus"

NEW PATH "exit"
ACTION $new_path "exit" 0

NEW_PATH "nav menus"
ACTION $new_path "nav_menus" 0

ACTION $host_sync "nav_menus" 1
    D 0 N
    N 2
       10 1 "H_" <esc> "[24;42H"
       24 14 26 "o continue, or ESC to exit"
       1 1 0
EDISPATCH
ACTION $host_send "nav_menus" 2
    N <cr>
ACTION $loop "nav_menus" 3
     Y 0 "item_transfer"

NEW_PATH "item_transfer"
```

```
ACTION $new_path "item_transfer" 0

ACTION $cl_clear "item_transfer" 1
     Y
ACTION $cl_move "item_transfer" 2
     1 1
ACTION $cl_send "item_transfer" 3
     N "item:"
ACTION $cl_get "item_transfer" 4
     item N 10
ACTION $loop "item_transfer" 5
     Y 0 "update_host"

NEW_PATH "update_host"
ACTION $new_path "update_host" 0

ACTION $cl_clear "update_host" 1
     Y
ACTION $cl_move "update_host" 2
     1 1
ACTION $cl_send "update_host" 3
     N "Processing Data"
ACTION $host_sync "update_host" 4
     D 0 N
     N 1
          10 2 "___" <esc> "[3;27H
          0  0  0
          0  0  0
EDISPATCH
ACTION $host_send "update_host" 5
     N item ;
     <cr>
ACTION $loop "update_host" 6
     Y 0 "item_error"

NEW_PATH "item_error"
ACTION $new_path "item_error" 0

ACTION $host_sync "item_error" 1
     D 0 N
     N 1
          10 1 "   " <^G><esc> "[24;79H"
          0  0  0
          0  0  0
EDISPATCH
ACTION $host_save "item_error" 2
     item_error 24 2 16
ACTION $cl_clear "item_error" 3
     Y
ACTION $cl_move "item_error" 4
     1 1
ACTION $cl_tone "item_error" 5
     Y
ACTION $cl_send "item_error" 6
     N item_error ;
     <cr> <lf> ;
```

```
        "Press ENTER to" ;
        <cr> <lf> '
        "continue:"
    ACTION $cl_get "item_error" 7
        answer N 32
    ACTION $host_sync "item_error" 8
        D 0 N
        N 1
            10 1 "   " <^G><esc> "[24;79H"
            0  0  0
            0  0  0
    ACTION $host_send "item_error" 9
        Y 0 "item_transfer"
    END_ACTION
    EOF
```

```
// test capture spec file
//

declare ans ""
declare foo ""
goto start

// ------------------------------------------------------------
// client connection

mark go_client
param $term_size 4,20
declare mode #2
if mode = "slave" then
connect remote pipe "/usr/pipes/fr_client"; "/usr/pipes/to_client"
else
connect remote tty "/dev/tty07"
configure remote "raw"
endif

display <cls> <cpos 2,4> "ScreenSheper"
display       <cpos 3,3> "Copyright 1994"
display       <cpos 4,1> "Orchid Systems, Inc."
display       <cpos 6,1>
return

// ------------------------------------------------------------
mark start
gosub go_client
declare_action #1

// during development: use "capture"
// during production: use "run_act!0^"
capture
// run_action

disconnect host
disconnect remote
eof
```

```
// designates a comment line


Declare storage variables

Bypass function definition(s)


Provide for connection to client
Replace with appropriate connect
for production version
Set client window size
Read command line argument "#2"
Slave connection is via unix Pipes

Otherwise use a default connection
to terminal no. 7
use in "raw" mode


Clear client screen and display a
standard startup message


return to caller of function


This is the actual start point
Open the client connection
Read in the PATHFILE specified on
the command line


Execute the capture function
or execute the run_act function

End connection to host
End connection to client
End the program execution
```

CLAIMS:

1. An apparatus for translating a first user interface from a preexisting application program running on a host computer to a second user interface running on a client computer comprising:

a computer adapted to monitor and capture interactions of a user using said application running on said host computer and further adapted to convert said interaction so that said interactions may be presented on said client computer;

first communication means for communicating between said computer and said host computer; and

second communication means for communicating between said computer and said client computer,

whereby said first user interface is modified for use on said second user interface.

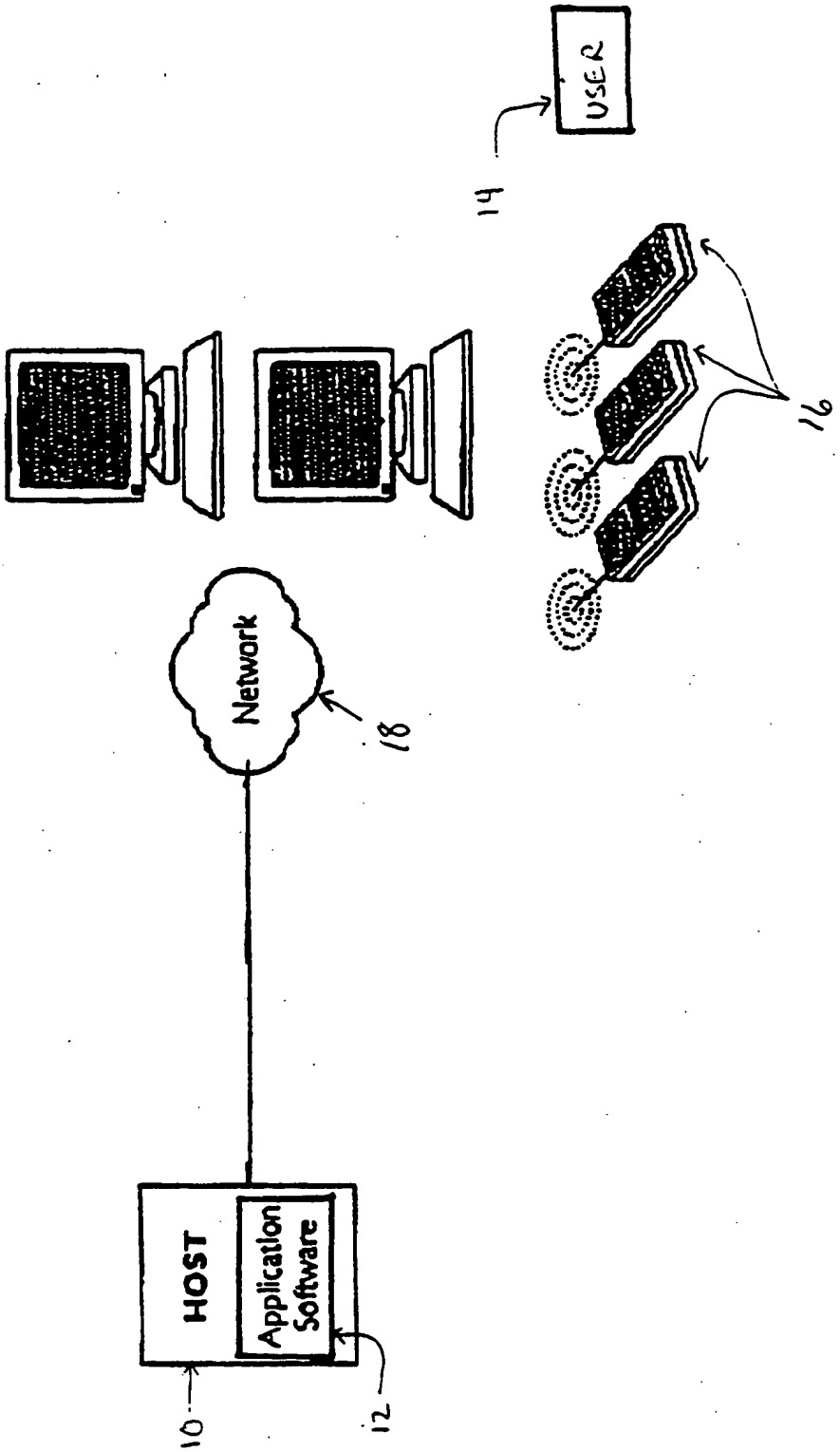


FIG. 1

2/75

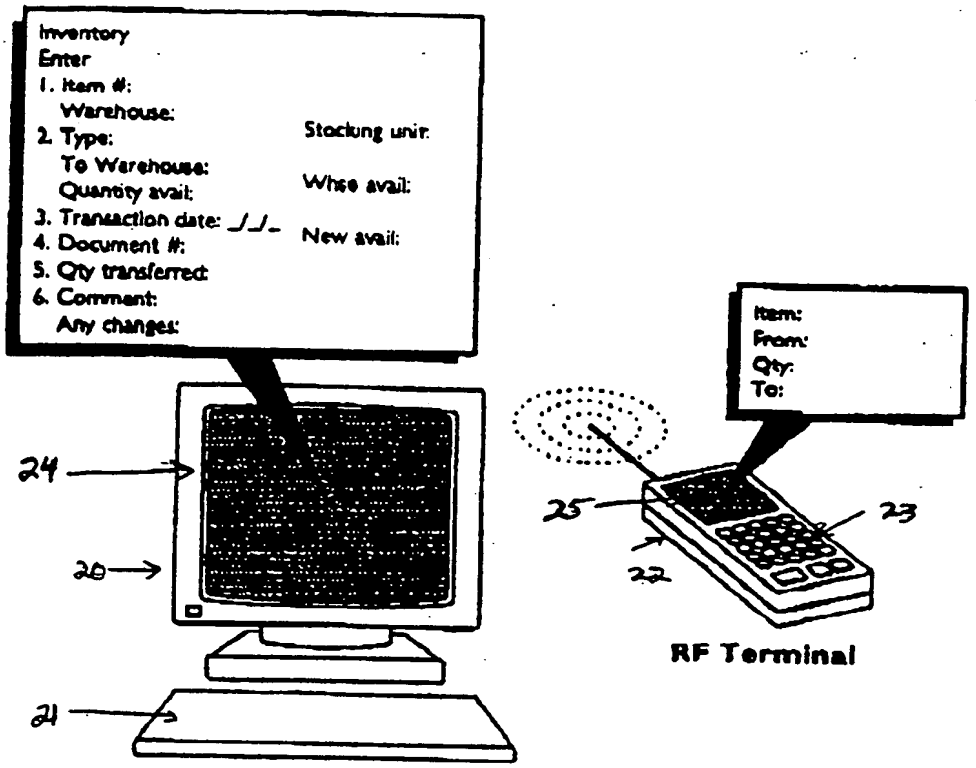
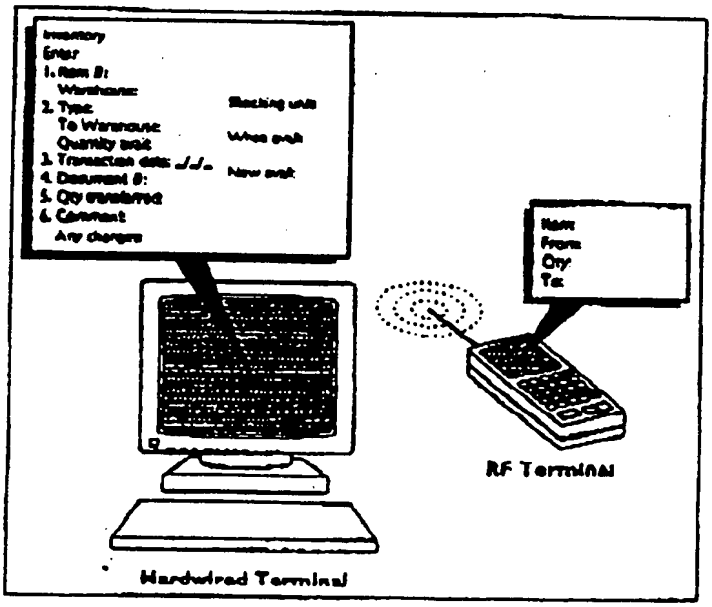
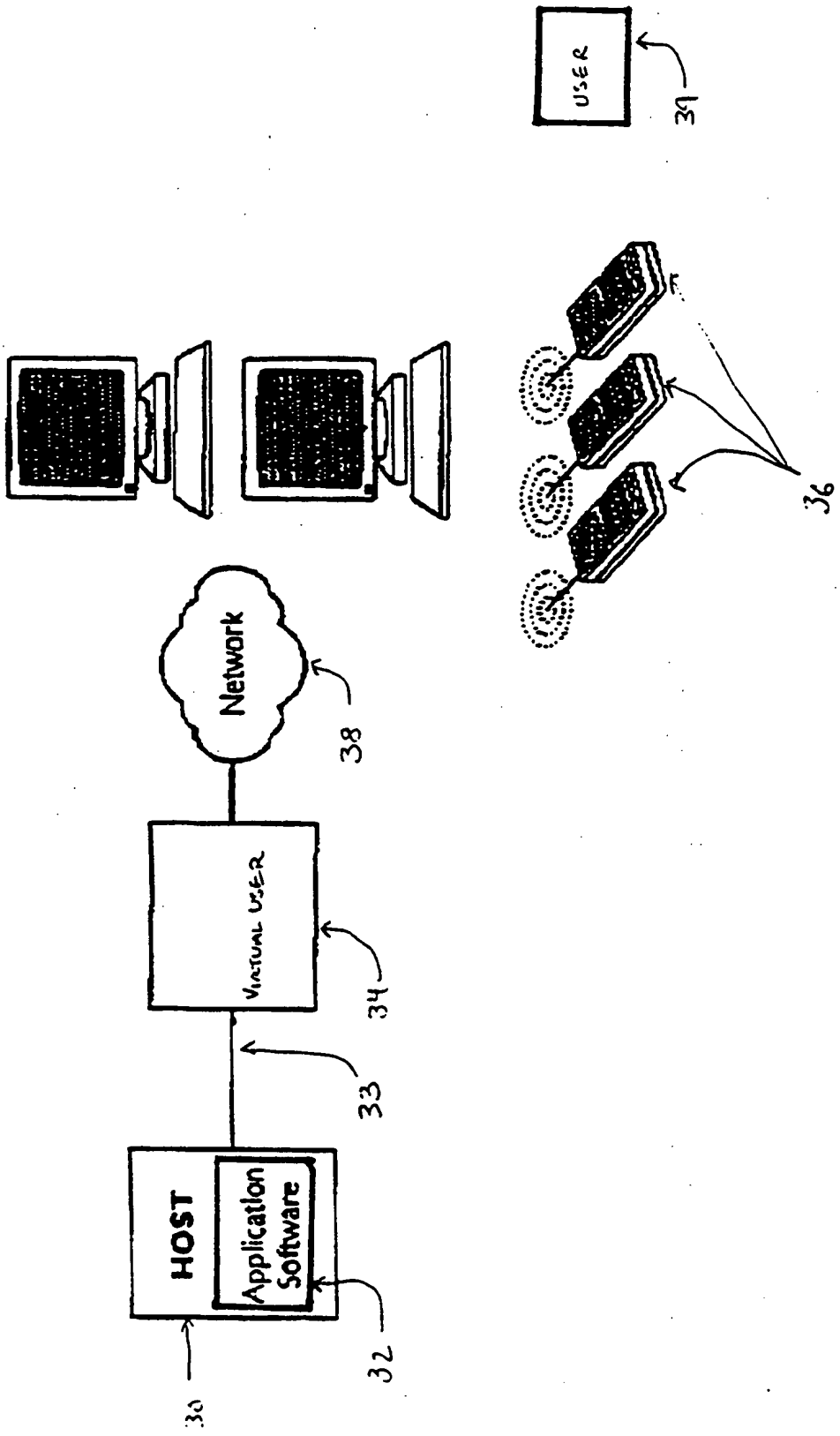

Inventory
Enter
1. Item #:
Warehouse:
2. Type:
To Warehouse:
Quantity avail:
3. Transaction date: _/_/_
4. Document #:
5. Qty transferred:
6. Comment:
Any changes:
Stocking unit:
Whse avail:
New avail:
24
20
21
Hardwired Terminal
Item:
From:
Qty:
To:
25
23
22
RF Terminal

Fig 2


RF Terminal
Hardwired Terminal


USER
39
36
Network
38
VIRTUAL USER
34
33
HOST
Application Software
32
30

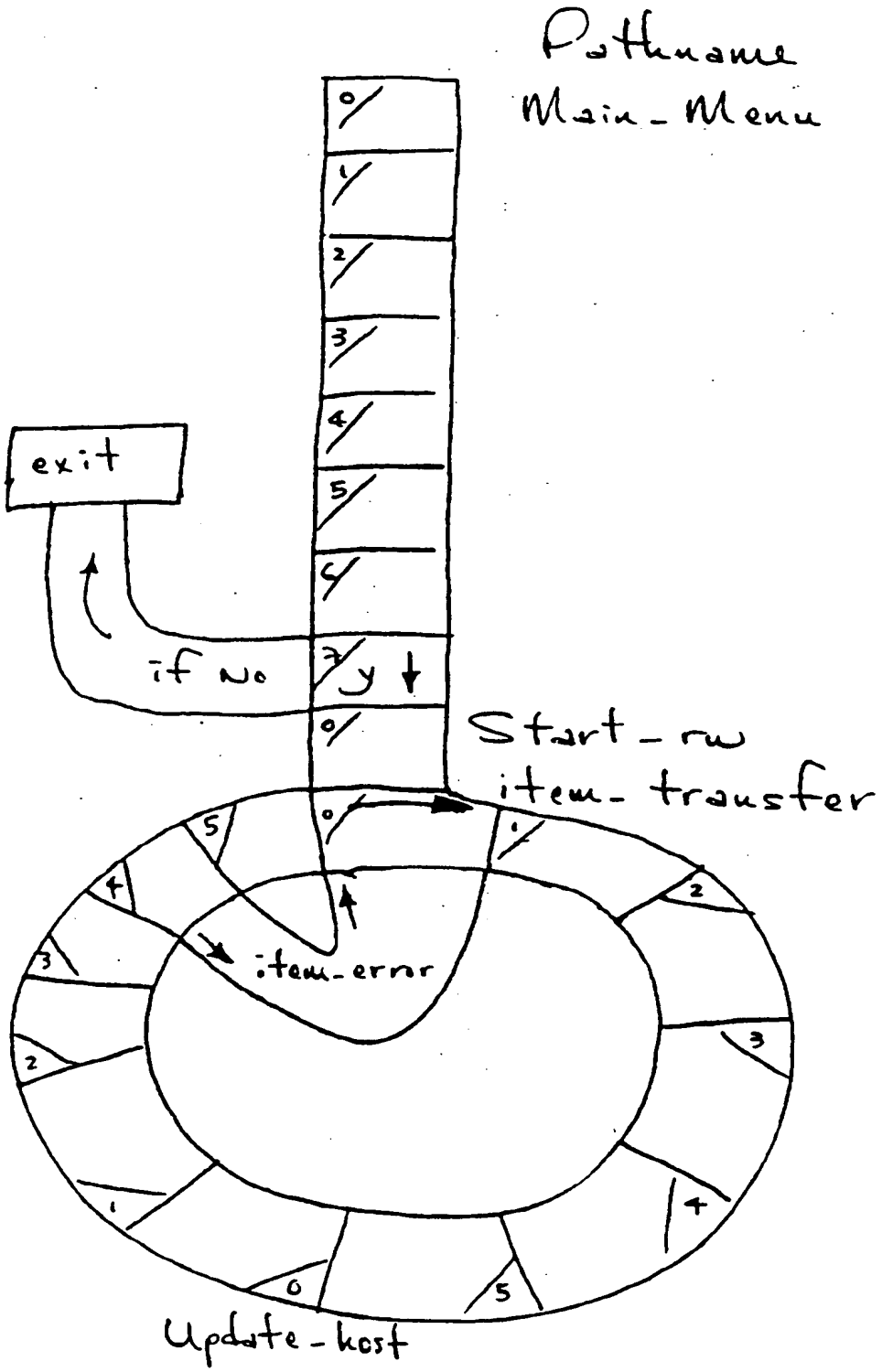
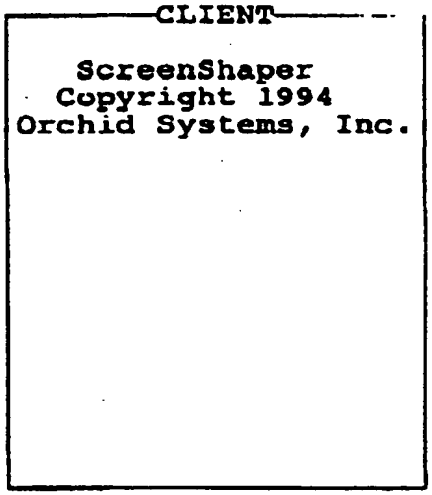

Pathname
Main_Menu
0
1
2
3
4
5
6
7
exit
if No
y
Start_rw
item_transfer
item_error
Update_host

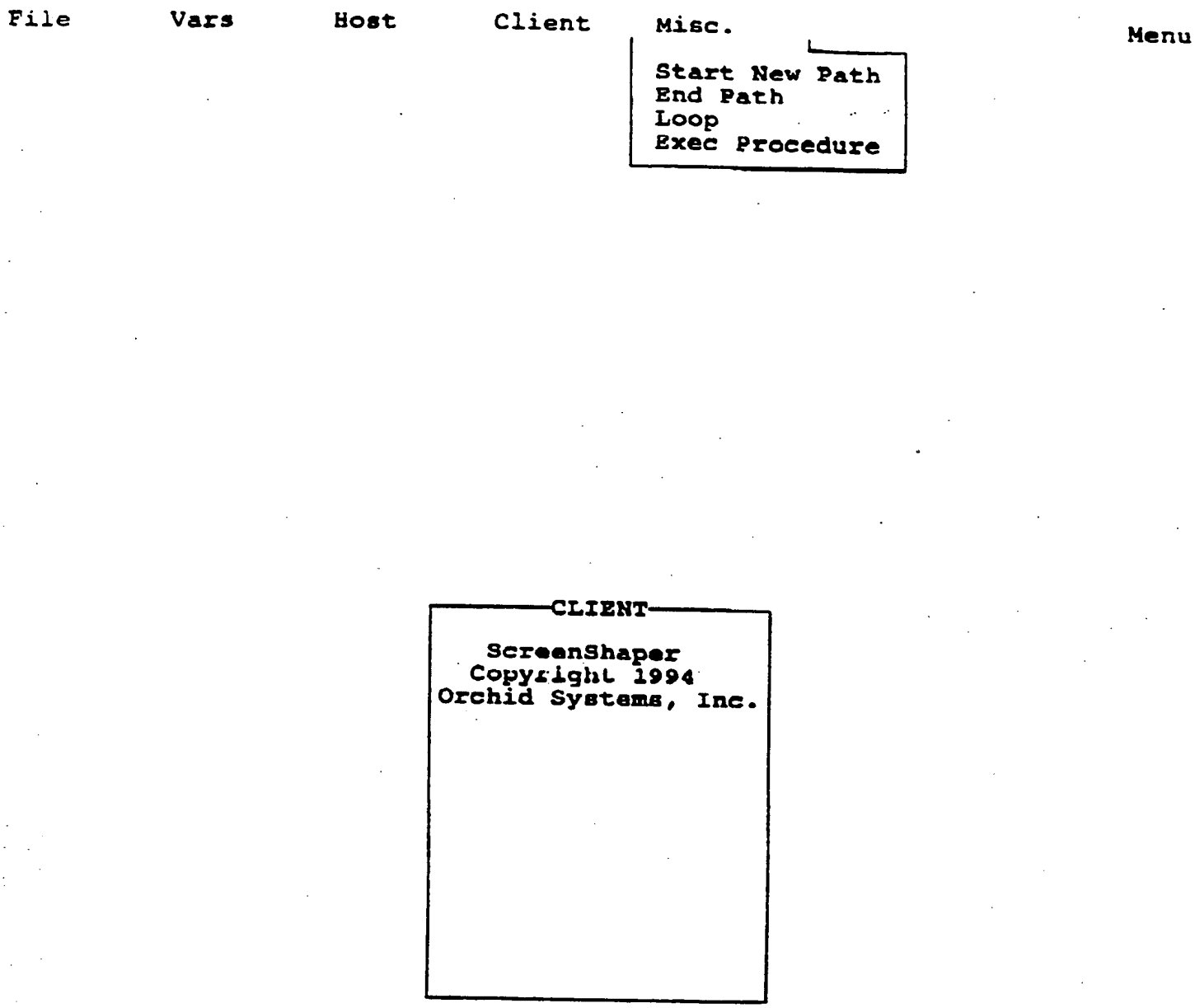
FIG. 4

File Vars Host Client Misc. Monitor



Section 3.3 - 1

Fig A-a



3.3-2 Step 0

Fig A-b

7/75

File Vars Host Cli nt Misc. Menu

Assign Path Name

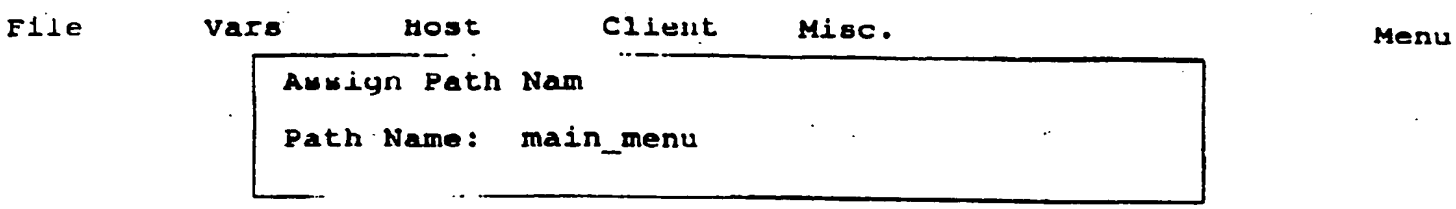
Path Name:

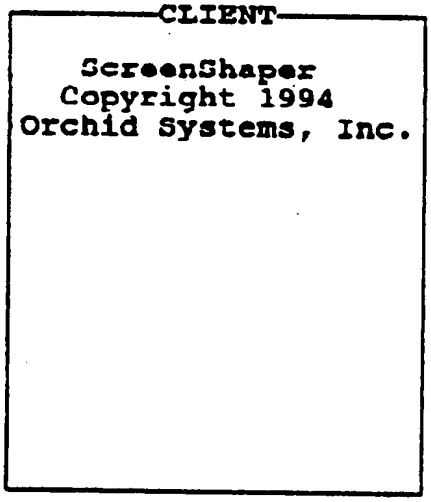
CLIENT

ScreenShaper
Copyright 1994
Orchid Systems, Inc.

33-2 Step 0

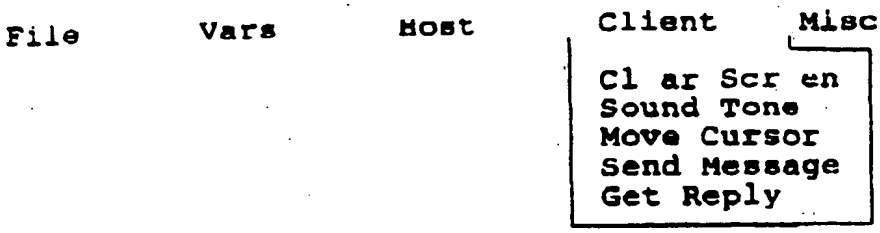
Fig A-C

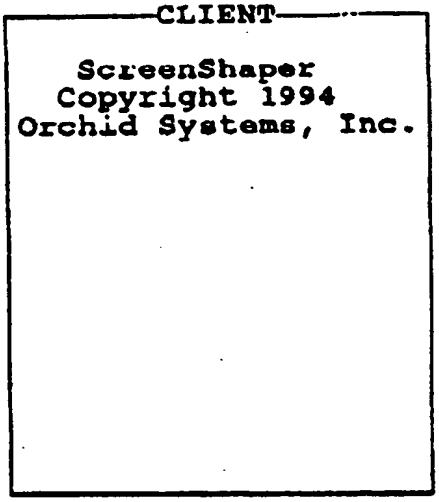




Step 0

Fig. A-d


File
Vars
Host
Client
Misc.
Menu
Clear Screen
Sound Tone
Move Cursor
Send Message
Get Reply


CLIENT
ScreenShaper
Copyright 1994
Orchid Systems, Inc.

Step 1

Fig 8-A

10/75

File Vars Host Client Misc. Menu

Move Client Cursor

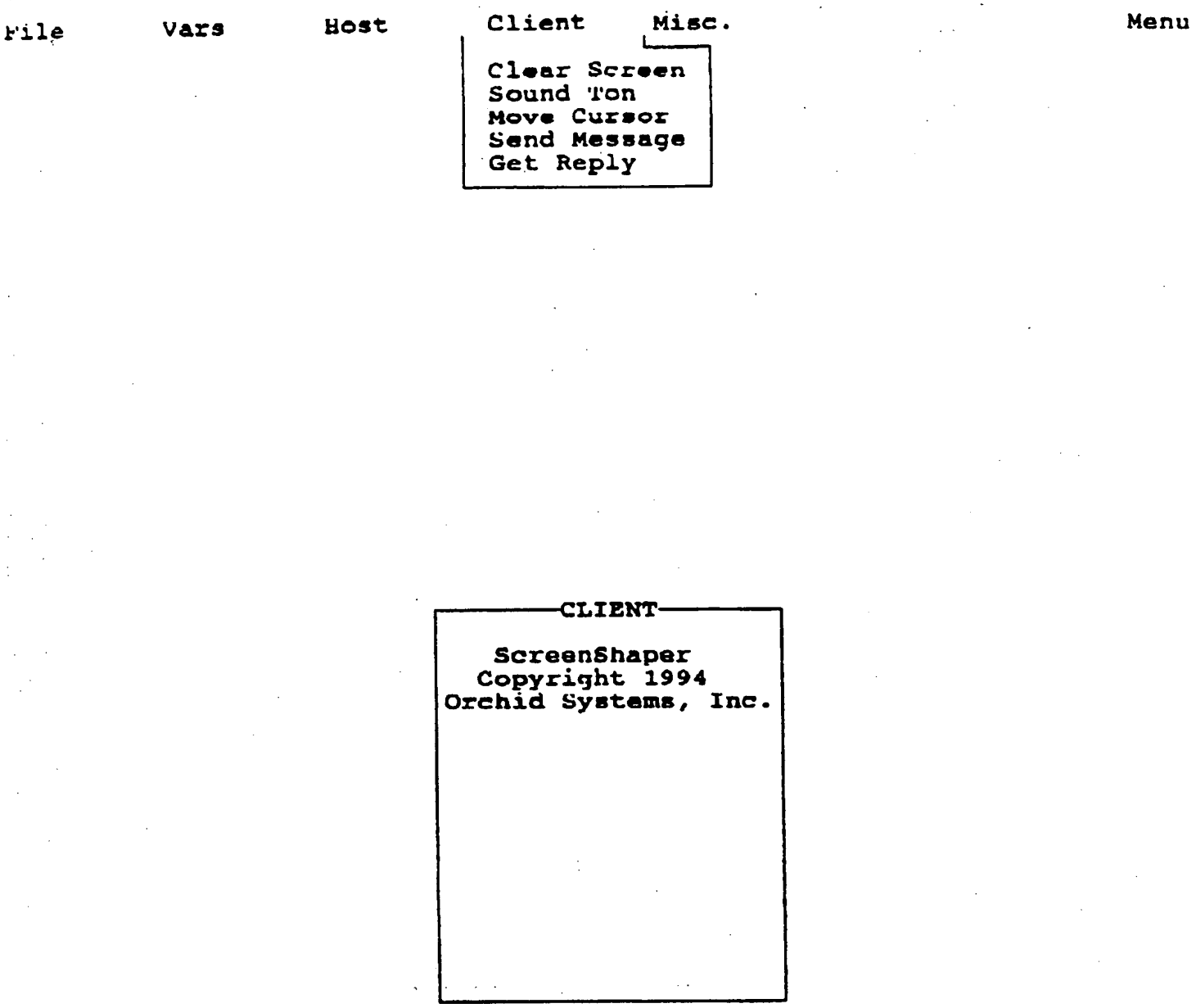
Row: 1
Col: 1

CLIENT

ScreenShaper
Copyright 1994
Orchid Systems, Inc.

Step 1

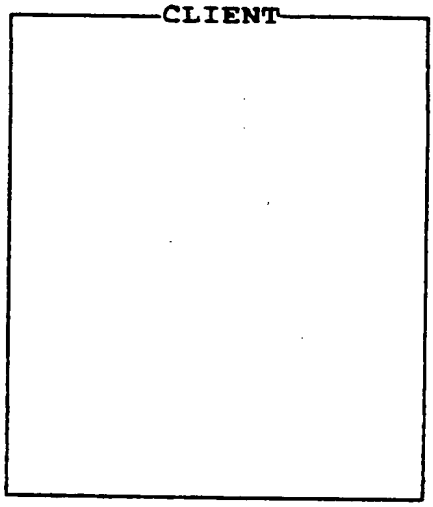
Fig B-B


File
Vars
Host
Client
Misc.
Menu
Clear Screen
Sound Ton
Move Cursor
Send Message
Get Reply
CLIENT
ScreenShaper
Copyright 1994
Orchid Systems, Inc.

Step 2

Fig 6-a

Fil Vars Host Client Misc. Monitor



Step B:

Fig C-b

File Vars Host Client Misc. Menu

Clear Screen
Sound Tone
Move Cursor
Send Message
Get Reply

CLIENT

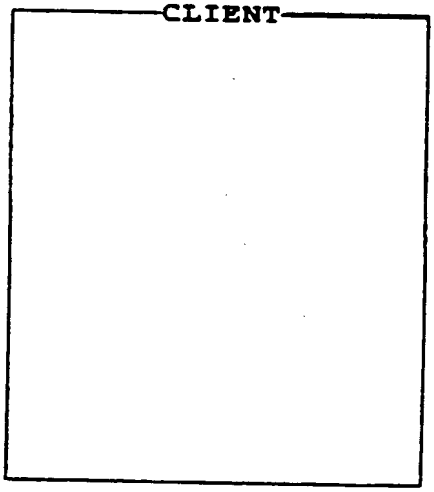
Step 3

Fig D-a

File Vars Host Client Misc. Menu

Send to Client

	Type	Value
Data 1:	<empty>	
Data 2:	<empty>	
Data 3:	<empty>	
Data 4:	<empty>	



Step 3

Fig 6-b

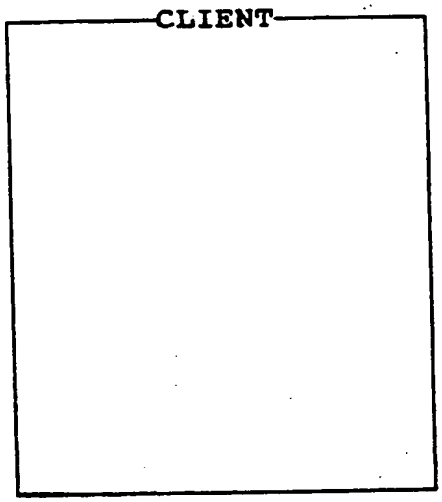
File Vars Host Client Misc. Menu

Send to Client

Data 1:
Data 2:
Data 3:
Data 4:

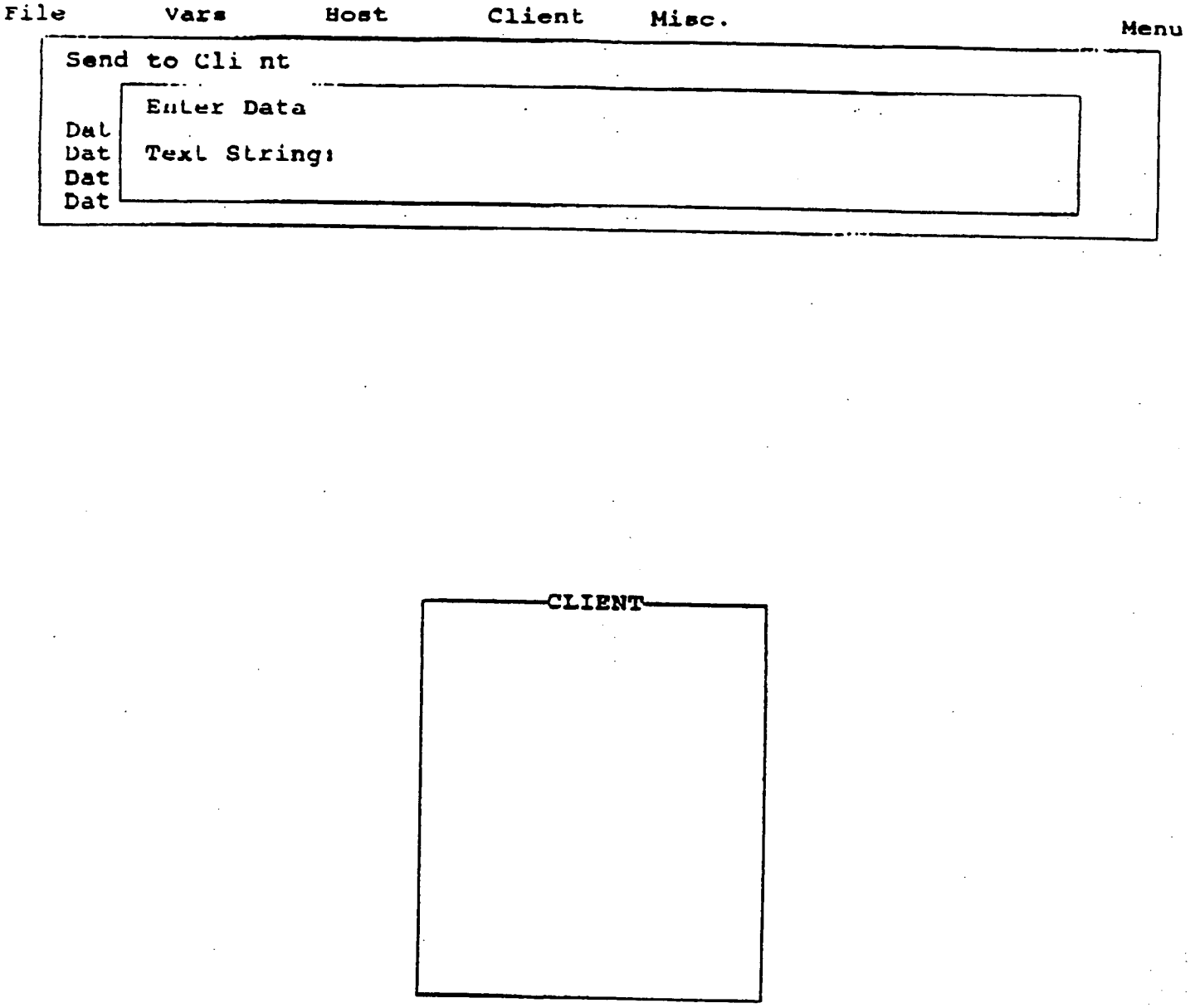
<empty>
Text
Special Char(s)
Variable

Value



Step 3

Fig 0-c


File
Vars
Host
Client
Misc.
Menu
Send to Cli nt
Enter Data
Dat
Dat
Dat
Dat
Text String:
CLIENT

Step 3

Fig 0-d

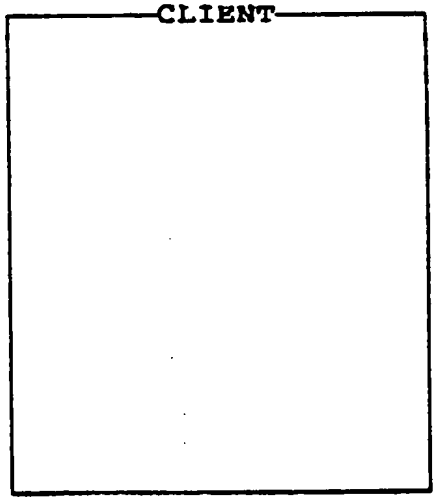
File Vars Host Client Misc. Menu

Send to Client

Data 1:
Data 2:
Data 3:
Data 4:

<empty>
Text
Special Char(s)
Variable

Value
Do you want to run.



Step 3

FIG 0-e

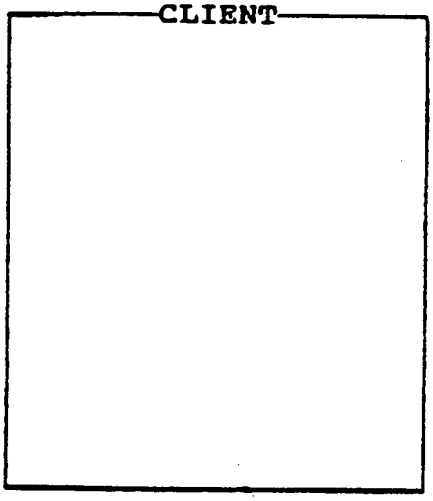
File Vars Host Client Misc. Menu

Send to Cli nt

Enter Data

Dat
Dat
Dat
Dat

Text String:the program (y/n):



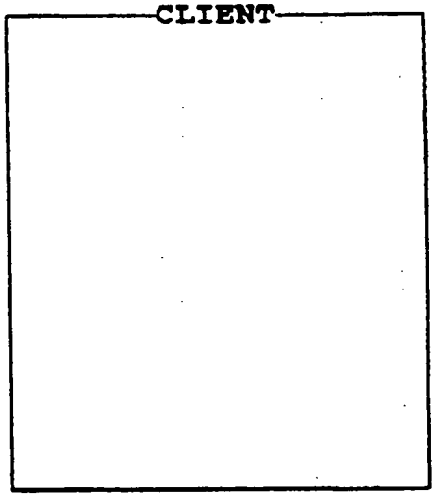
Step 3

Fig 0-f

File Vars Host Client Misc. Menu

Send to Client

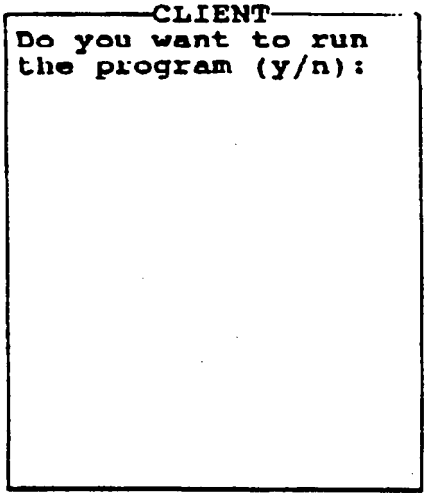
	Type	Value
Data 1:	Text	Do you want to run
Data 2:	Special Char(s)	<cr> <lf>
Data 3:	Text	the program (y/n):
Data 4:	<empty>	



Step 3

FIG 9-g

Fil Vars H st Client Misc. Monitor



Step 3

Fig D-h

File Vars Host Client Misc. Menu

Declare Variable
Set Variable
Branch on Variable

CLIENT
Do you want to run the program (y/n):

Step 4

Fig E-a

File Vars Host Client Misc. Menu

Declare New Variable

Variable:

CLIENT

Do you want to run the program (y/n):

Step 4

Fig E-b

File Vars Host Client Misc. Menu

Declare New Variable

Variable: answer

CLIENT

Do you want to run the program (y/n):

Step 9

Fig E-C

File Vars Host Client Misc. Menu

Clear Scr en
Sound Ton
Move Cursor
Send Message
Get Reply

CLIENT
Do you want to run the program (y/n):

Step 5

Fig 5a

File Vars Host Client Misc. Menu

G t Client R ply

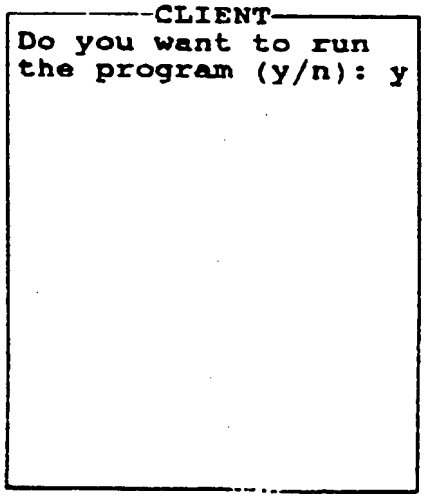
Variable Name: answer
Max Len: 1

CLIENT
Do you want to run the program (y/n):

Step 5

Fig F-b

File Vars Host Client Misc. Monitor



Step 6
Type "y" into
Client screen

FIG 6

File Vars Host Client Misc. Menu

Declare Variable
Set Variabl
Branch on Variable

CLIENT
Do you want to run
the program (y/n): y

Step 7

Fig H-9

File Vars Host Client Misc. Menu

Dispatch on Variable

Variable: answer

	Value to Match	Path Name	No.
1.	eq y	start_rw	0
2.	eq		0
3.	eq		0
4.	eq		0
not.	eq		0

CLIENT

Do you want to run
the program (y/n):y

Step 7

FIG. 4-b

25

File Vars Host Cli nt Misc. M nu

Di Path start_rw do sn't exist

Va D. Declare Pathname
R. Respecify Pathname
C. Cancel Operation

Select [D,R,C]: R

No.
1. 0
2. 0
3. 0
4. eq 0
not eq 0

CLIENT
Do you want to run
the program (y/n):y

Step 7

Fig 4-c

File Vars Host Client Misc. Menu

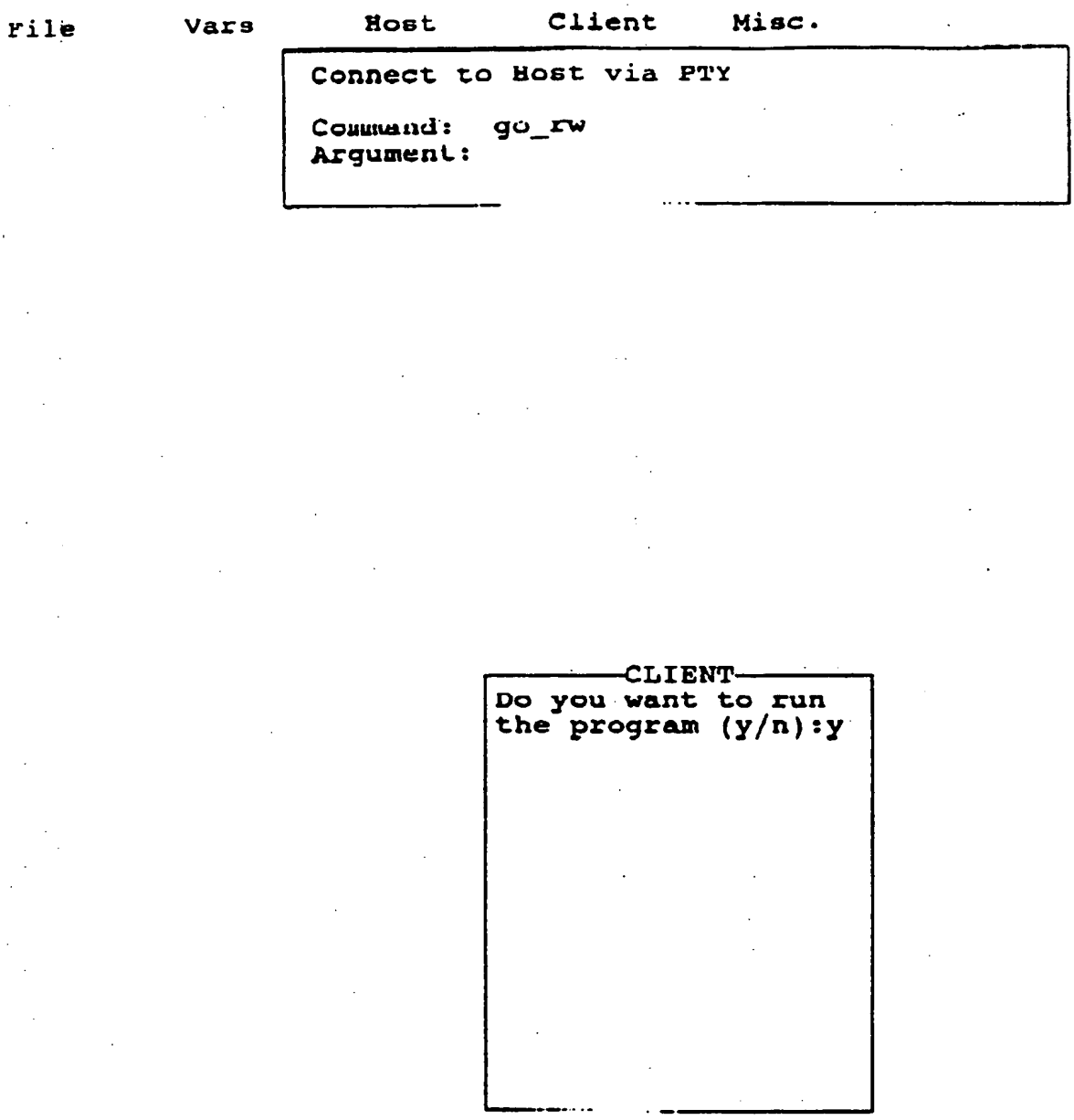
Connect Local (PTY)
Connect Local (Pipe)
Connect via TELNET
Connect via Serial

CLIENT

Do you want to run
the program (y/n):y

3 Step 0

Fig I-a



Step 0

Fig I-b

12:

File Vars Host Client Misc. Monitor

HOST

RealWorld Software
Version 6.5
Data Look Up Utility 3.0 (C) 1992, 1993 by DataTech
(C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation.

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 0

Fig I-c

29

File Vars Host Client Misc. Menu

Start New Path
End Path
Loop
Exec Procedure

HOST

RealWorld Software
Version 6.5
Data Look Up Utility 3.0 (C) 1992, 1993 by DataTech
(C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation

ALL RIGHTS RESERVED. No part of this material may be reproduced or transmitted in any form or by any means, electronic, mechanical or otherwise, without written permission from RealWorld Corporation.

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, N w Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 0

Fig 3-a

File Vars Host Client Misc. Menu

Current Path start_rw not ended.

T. Terminate Path
E. Link to Existing Path
N. Link to New Path
C. Cancel Add Path

Select [T,E,N,C] N

RealWorld Sof
Version 6.5
Data Look Up Utility 3.0 (C) 1992, 1993 by DataTech
(C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such licens , do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 0

Fig 3-b

File Vars Host Cli nt Misc. Menu

Assign Path Nam

Path Name: nav_menus

HOST

RealWorld Software
Version 6.5
Data Look Up Utility 3.0 (C) 1992, 1993 by DataTech
(C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation.

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 0

Fig 3-c

File Vars Host Client Misc. Menu

Send to Host
Wait for Host
Save Host Data
Bypass Virt. User
Configure Emulator
Disconnect Host

HOST

RealWorld Software
Version 6.5
Data Look Up Utility 3.0 (C) 1992, 1993 by DataTech
(C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 1

FIG K-9

34

File Vars Host Client Misc. Menu

Wait for Host Respons (6)

Possible Response	-- unused -- automatic screen match	Destination Path	No.
0. Normal Response			0
1.	-- unused --		0
2.	-- unused --		0
3.	-- unused --		0
4.	-- unused --		0
5.	-- unused --		0
6. None of the Above	--- None ---		0

Wait a maximum of 0 x .1 sec

RealWo ation
V rsio
Data L
(C) Co

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 1

Fig K-b

File Vars Host Client Misc. Menu

Define Scr en Match

	Loc	Len	Expected
Wait For:		10	1 x [B_\e[24;42H]
Match at:	24,14	26	[o continue, or ESC to exi
Match at:	1, 1	0	[]

RealWorld Softw
Version 6.5
Data Book Up Ut
(C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation

ALL RIGHTS RESERVED. No part of this material may be reproduced or transmitt d in any form or by any means, electronic, mechanical or otherwis , without written permission from RealWorld Corporation.

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Pr ss ENTER to continue, or ESC to exit _

Step 1

FIG K-C

36

34/75

File Vars Host Client Misc. Menu

Send to Host
Wait for Host
Save Host Data
Bypass Virt. User
Configure Emulator
Disconnect Host

HOST

RealWorld Software
Version 6.5
Data Look Up Utility 3.0 (C) 1992, 1993 by DataTech
(C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation.

ALL RIGHTS RESERVED. No part of this material may be reproduced or transmitted in any form or by any means, electronic, mechanical or otherwis , without written permission from RealWorld Corporation.

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such licens , do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 2

Fig 1-a

37

File Vars Host Client Misc. Menu

Send to Hu
< mpty>
Text
Special Char(s)
Variable
<empty>

Value

Data 1:
Data 2:
Data 3:
Data 4:

RealWorld Software
Version 6.5
Data Look Up Utility 3.0 (C) 1992, 1993 by DataTech

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 2.

FIG 1-b

35

41/75

File Vars Host Client Misc. Menu

Send to Host

	Type	Value
Data 1:	Special Char(s)	<cr>
Data 2:	<empty>	
Data 3:	<empty>	
Data 4:	<empty>	

Realw rld Software
Version 6.5
Data Look Up Utility 3.0 (C) 1992, 1993 by DataTech
(C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If you need another copy of the license, contact your supplier or RealWorld Corporation at Loudon Road, Concord, New Hampshire 03301 USA. (Telephone 800-678-6336.)

Press ENTER to continue, or ESC to exit _

Step 2

Fig 1-c

39

42/75

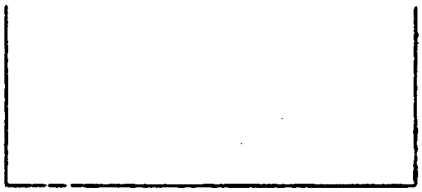
File vars Host Cli nt Misc. Monitor

HOST

ealWorld Software MSI
ersion 6.5

Please enter your initials: ___

ress F1 for version numbers
ress F2 for forms ordering info



Step 3

FIG M

40

43/75

File Vars Host Client Misc. Monitor

HOST

nventory MSI
nter
1. Item #
Warehouse

2. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

1 = next entry F2 = next item blank = look up by description

item_transfer
step 0

Fig N-a

File Vars Host Cli nt Misc. Menu

Current Path start_rw not ended.

T. Terminate Path
E. Link to Existing Path
N. Link to New Path
C. Cancel Add Path

Select [T,E,N,C] N

nventory MSI
nter
1. Item #
Warehouse

2. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

1 = next entry F2 = next item blank = look up by description

Fig N-b

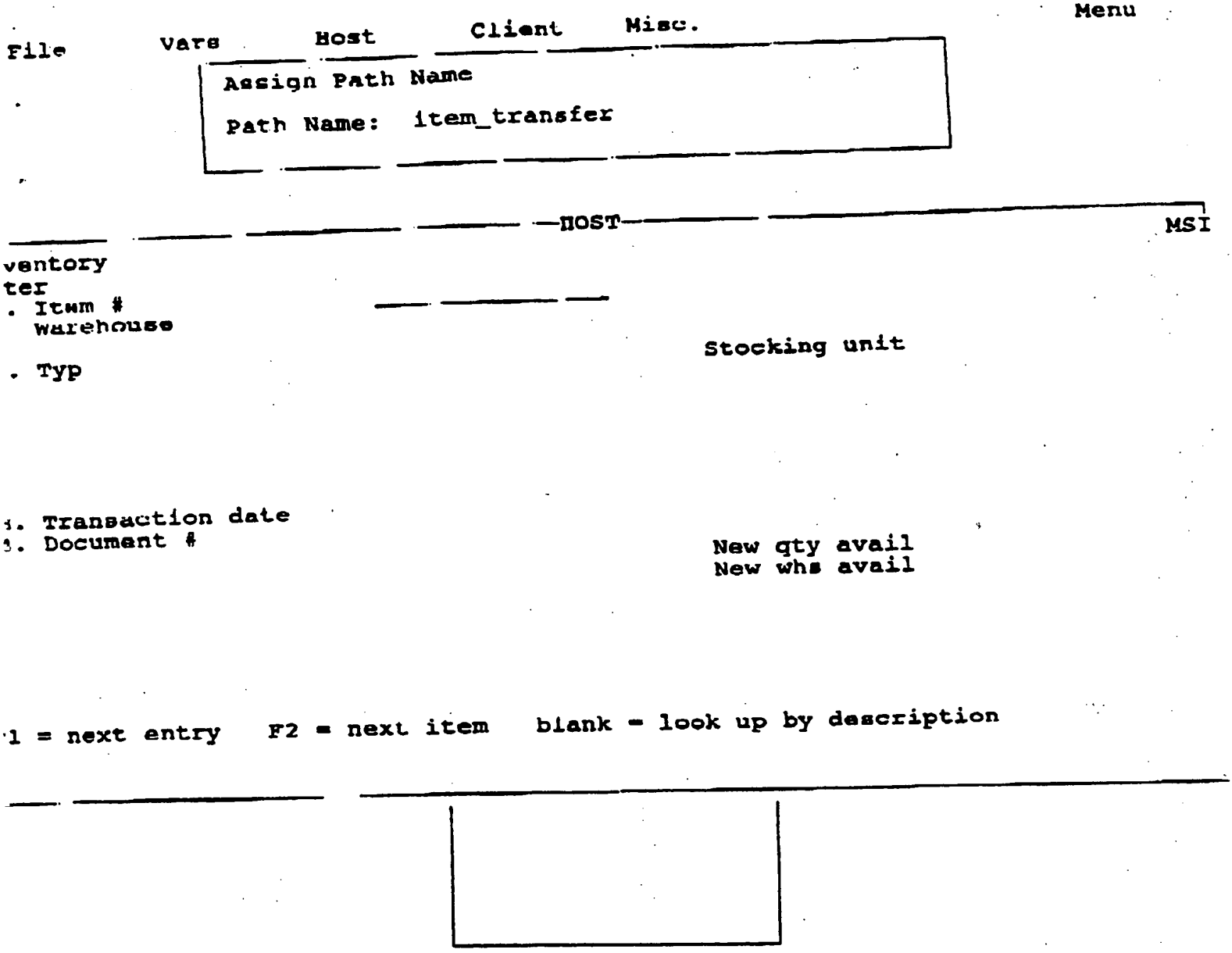

Menu
File
Vars
Host
Client
Misc.
Assign Path Name
Path Name: item_transfer
HOST
MSI
ventory
ter
. Item #
warehouse
Stocking unit
. Typ
. Transaction date
. Document #
New qty avail
New whs avail
1 = next entry
F2 = next item
blank = look up by description

FIG N-C

File Vars Host Client Misc. Menu

Clear Client Screen

Enable: Y

HOST

nventory MSI
:nter
1. Item #
Warehouse

2. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

F1 = next entry F2 = next item blank = look up by description

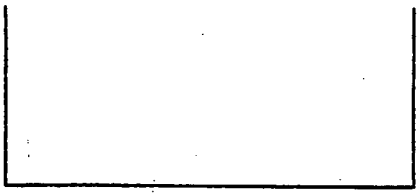


Fig 0

47/75

File Vars Host Client Misc. Menu

Move Client Cursor

Row: 1
Col: 1

HOST MSI

nventory
nt r
1. Item #
Warehouse

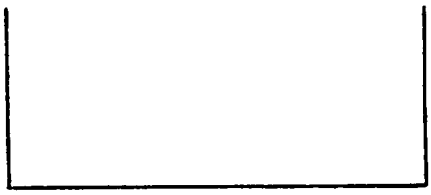
2. Type

Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

F1 = next entry F2 = next item blank = look up by description



2.

Fig 9

File Vars Host Client Misc. M nu

Send to Client

	<empty>	Value
Data 1:	Text	
Data 2:	Special Char(s)	
Data 3:	Variable	
Data 4:		

nventory MSI
nter
1. Item #
Warehouse

2. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

F1 = next entry F2 = next item blank = look up by description

3

Fig 9-a

File Vars Host Client Misc. Menu

Send to Client

	Type	Value item:
Data 1:	Text	
Data 2:	<empty>	
Data 3:	<empty>	
Data 4:	<empty>	

MSI

:aventory
nter
1. Item #
Warehouse

2. Type

Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

:1 = n xt entry F2 = next item blank = look up by description

Fig 6-b

File Vars Host Client Misc. Menu

Declare New Variable

Variable: item

HOST

MSI

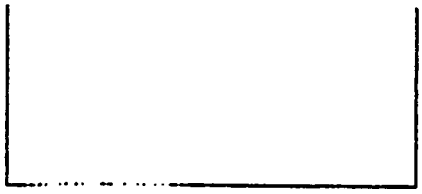
:nventory
nter
1. Item #
Warehouse

2. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

1 = next entry F2 = next item blank = look up by description



4

FIG R

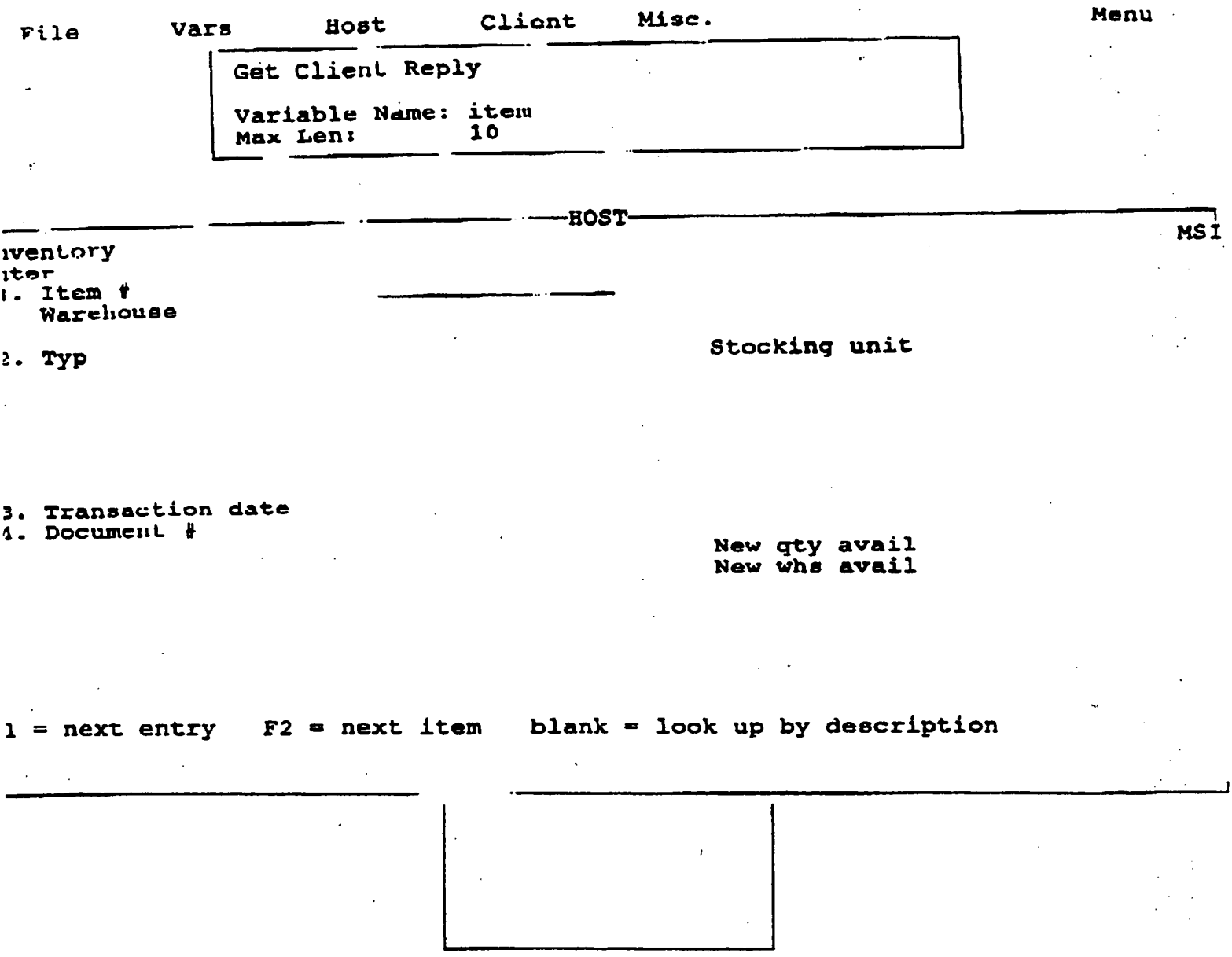

File
Vars
Host
Client
Misc.
Menu
Get Client Reply
Variable Name: item
Max Len: 10
HOST
MSI
ıventory
ıter
1. Item #
Warehouse
2. Typ
Stocking unit
3. Transaction date
4. Document #
New qty avail
New whs avail
1 = next entry F2 = next item blank = look up by description

Fig 5-a

File Vars Host Client Misc. Monitor

HOST

nventory MSI
nter
1. Item #
Warehouse

2. Typ Stocking unit

3. Transaction date
4. Document #

New qty avail
whs avail

CLIENT
item:1

F1 = next entry F2 = next i by description

Note: "1" is a valid item number

FIG 5-b

5

File Vars Host Client Misc. Menu

Assign Path Name

Path Name: update_host

HOST MSI

iv ntory
iter
.. Item #
Warehouse

2. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

1 = n xt entry F2 = next item blank = look up by description

update_host

Step 0

Fig 7

File Vars Host Client Misc. Menu

Clear Client Screen

Enable: Y

HOST

MSI

Inventory
Item
.. Item #
Warehouse

:. Type

Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

= next entry F2 = next item blank = look up by description

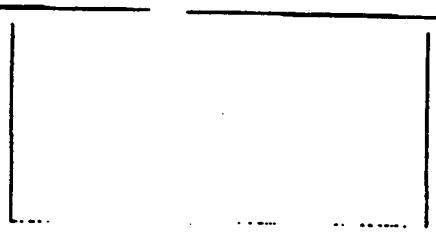


Fig 4

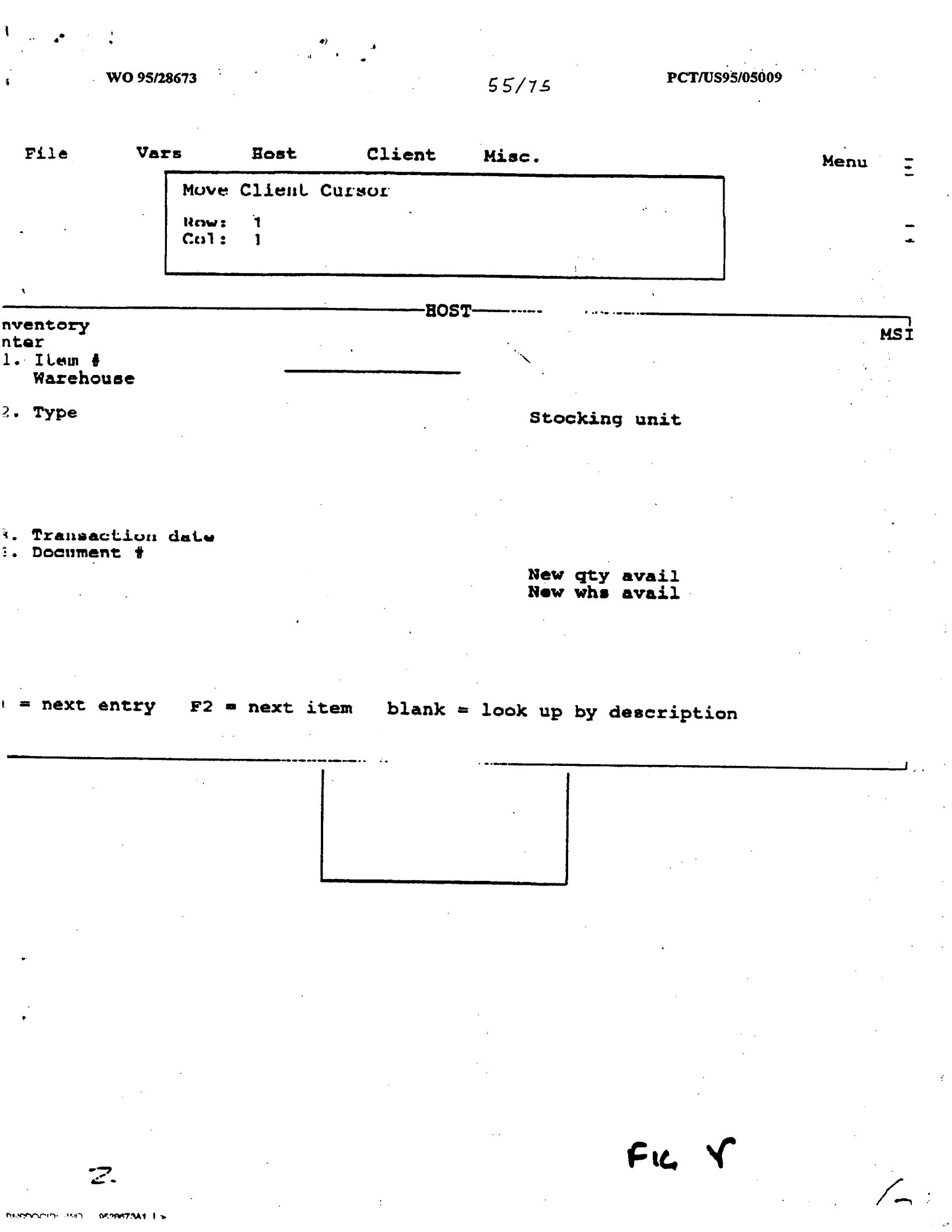

File
Vars
Host
Client
Misc.
Menu
Move Client Cursor
Row: 1
Col: 1
HOST
nventory
nter
MSI
1. Item #
Warehouse
2. Type
Stocking unit
3. Transaction date
4. Document #
New qty avail
New whs avail
= next entry
F2 = next item
blank = look up by description

Fig 7

2.

File Vars Host Client Misc. Menu

Send to Cli nt

	Type	Value
Data 1:	Text	Processing Data
Data 2:	<empty>	
Data 3:	<empty>	
Data 4:	<empty>	

ventory
ter MSI
. It m #
Warehouse

. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

= next entry F2 = next item blank = look up by description

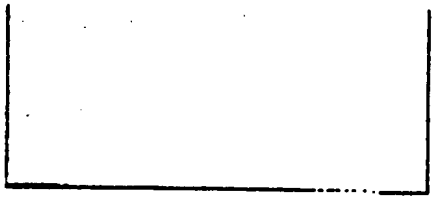


FIG W

File Vars Host Client Misc. Menu

Wait for Host Response(s)

Possible Response	Test Type	Destination Path	No.
0. Normal Response	automatic		
1.	-- unused --		0
2.	-- unused --		0
3.	-- unused --		0
4.	-- unused --		0
5.	-- unused --		0
6. None of the Above	--- None ---	item_error	0

Wait a maximum of 100 x .1 sec

nvent
nter
1. It
Wa
2. Ty

MSI

3. Transaction date
4. Document #

New qty avail
New whs avail

eav blank for "Central"

Pathname Update - host

Fig X-a

Note: Error handling will be provided in the None of the above path "item_error"

Time out set for 100x.1 = 10 sec

Step 4

15

File Vars Host Client Misc. Menu

Wait fo
Path it m_error doesn't exist
Poss
D. Declare Pathname
R. Respecify Pathname
C. Cancel Operation
0. Norm
1.
2.
3.
4.
5.
6. None of the Above
Select [D,R,C]: R
-- unused --
-- unused --
--- None --- item_error
Wait a maximum of 0 x .1 sec

No.
0
0
0
0
0
0

MSI

ivent
ilter
1. It
Wa
2. Ty

3. Transaction date
1. Document #

New qty avail
New whs avail

eave blank for "Central"

Fig X-b

4

File Vars Host Client Misc. Menu

Send to Host

	Type	Value
Data 1:	Variable	item
Data 2:	Special Char(s)	<cr>
Data 3:	<empty>	
Data 4:	<empty>	

ıventory
ıter MSI
. Item #
Warehouse

. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

l = next ntry F2 = next item blank = look up by description

Fig 4-a

5

/12

File Vars Host Client Misc. Monitor

HOST

iv nt ry MSI
ıLer
.. ltem # 1 Paint, Black
Warehouse

'. Type Stocking unit EACH

i. Transaction date
i. Document #

New qty avail
New whs avail

:ave blank for "Central"

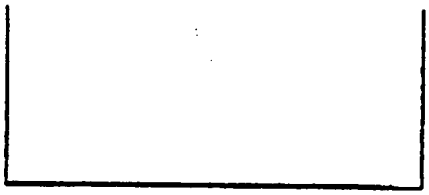


Fig 4-b

Fil Vars Host Client Misc. Monitor

HOST

ealWorld Software
ersion 6.5
ata Look Up Utility 3.0 (C) 1992, 1993 by DataTech
C) Copyright 1986, 1987, 1988, 1989, 1990, 1991, 1992 by RealWorld Corporation.

ALL RIGHTS RESERVED. No part of this material may be reproduced or transmitted in any form or by any means, electronic, mechanical or otherwise, without written permission from RealWorld Corporation.

This software may be used only as authorized by a valid "RealWorld Software License". If you have not read and agreed to such license, do not continue. If yo the license, contact your supplier or RealWo don Road, Concord, New Hampshire 03301 USA 6336.)

CLIENT
item:15

ress ENTER to continue, or E

Second time through we enter invalid data "15" and "train" Virtual User how to respond

FIG 2-9

File Vars Host Client Misc. Menu

Wait for Host Respons (s)

Possible Response	Test Type	Destination Path	No.
0. Normal Response	automatic		
1.	-- unused --		0
2.	-- unused --		0
3.	-- unused --		0
4.	-- unused --		0
5.	-- unused --		0
6. None of the Above	--- None ---		0

Wait a maximum of 0 x .1 sec

nvent
nter
1. It
Wa
2. Ty

MSI

3. Transaction date
4. Document #

New qty avail
New whs avail

It m not on file

Press ENTER or F8

Pathname item_error

Step 0

Fig 2-b

21

63/75

File Vars Host Client Misc. Menu

Declare New Variable

Variable: item_error

HOST

ventory MSI
ter
. Item # 15
Warehouse

. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

tem not on file Press ENTER or F8

Fig AA

22

64/75

File Vars Host Client Misc. Menu

Save Host Screen Data

Variable: item_error

	Loc	Len	Expected
Save at:	24, 2	16	[Item not on file]

HOST

ventory
ter MSI
. Item # 15
Warehouse

. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

tem not on file]

Press ENTER or F8

error window

Note: Captures any data in error window

2

Fig 8B

12

65/75

File Vars Host Client Misc. Menu

Clear Client Screen

Enable: Y

HOST

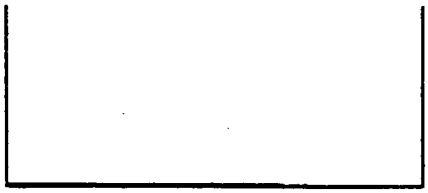
ventory MSI
er
. Item # 15
Warehouse

. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

em not on file Press ENTER or F8



3 Fig cc 124

66/75

File Vars Host Client Misc. Menu

Move Client Cursor

Row: 1
Col: 1

HOST

nv ntory
:nter MS
1. Item # 15
Warehouse

2. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

Item not on file

Press ENTER or F8

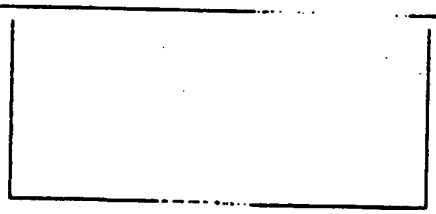


Fig DD

File Vars Host Client Misc. Menu

Sound Tone at Client

Enable: Y

HOST

v ntory MSI
ter
. Item # 15
Warehouse

. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

tem not on file Press ENTER or F8

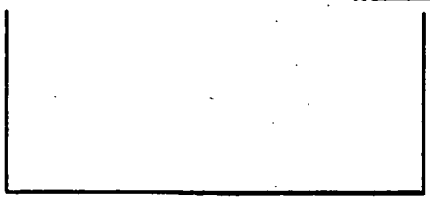


FIG EE

File Vars Host Clicnt Misc. Menu

Send to Client

	Type	Value
Data 1:	Variable	item_error
Data 2:	Special Char(s)	<cr> <lf>
Data 3:	Text	Press ENTER to
Data 4:	Special Char(s)	<cr> <lf> "continue:"

Inventory
Iter MSI
.. Item # 15
Warehouse

. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

tem not on file Press ENTER or F8

6

FIG FF

27

File Vars Host Client Misc. Menu

Declare New Variable

Variable: answer

HOST

ventory MSI
ler
. Item # 15
Warehouse
. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

t m not on file Press ENTER or F8

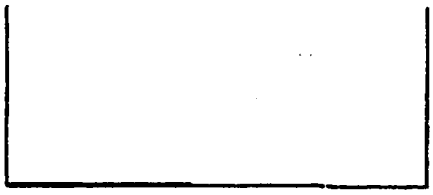


Fig 6G

File Vars H st Client Misc. Menu

Get Client R ply

Variable Name: answer
Max Len: 3

HOST

MSI

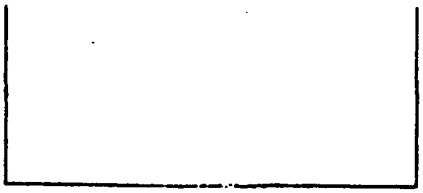
ventory
ter
.. Item # 15
Warehouse

:. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

Item not on file Press ENTER or F8



8

Fig HH

29

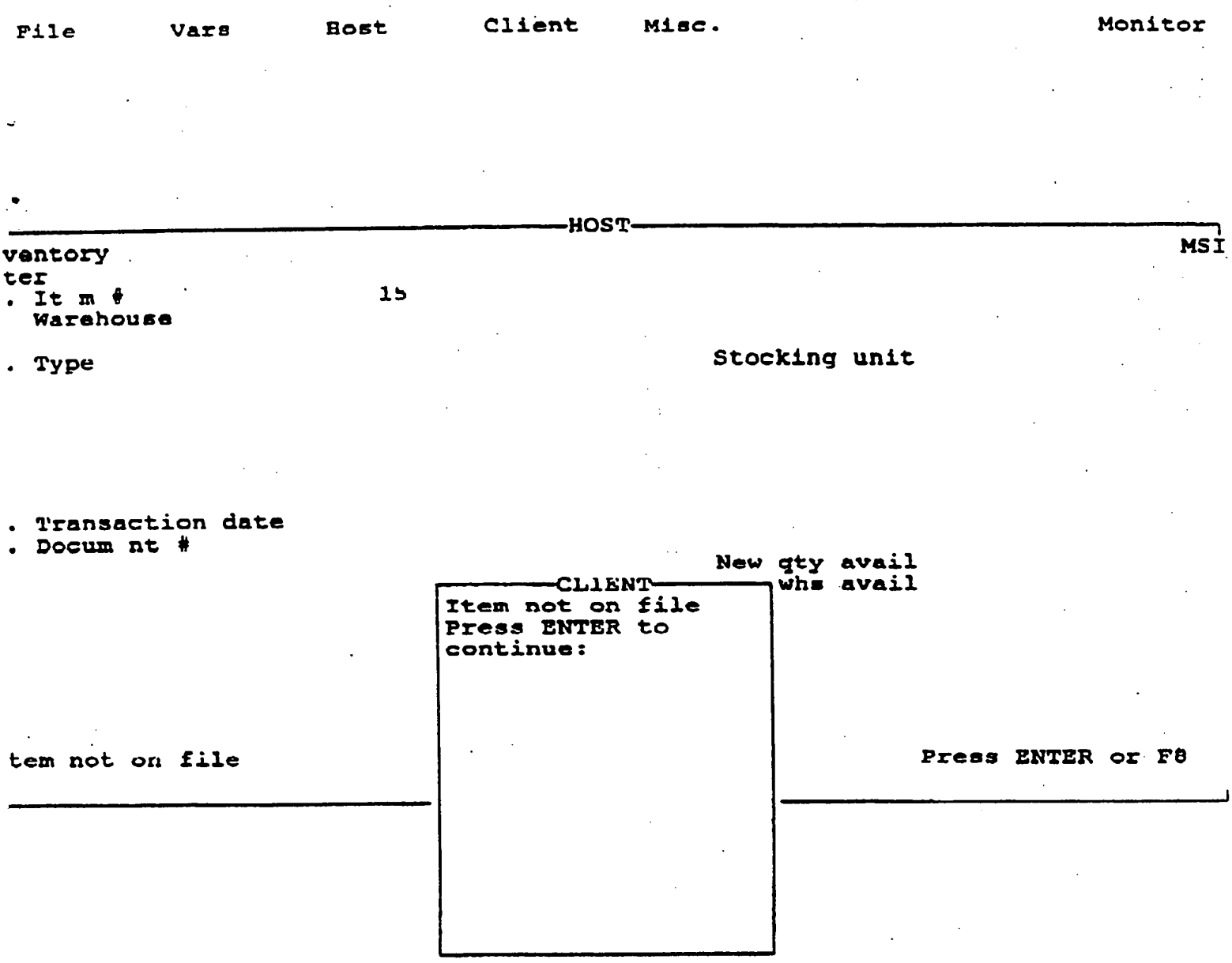
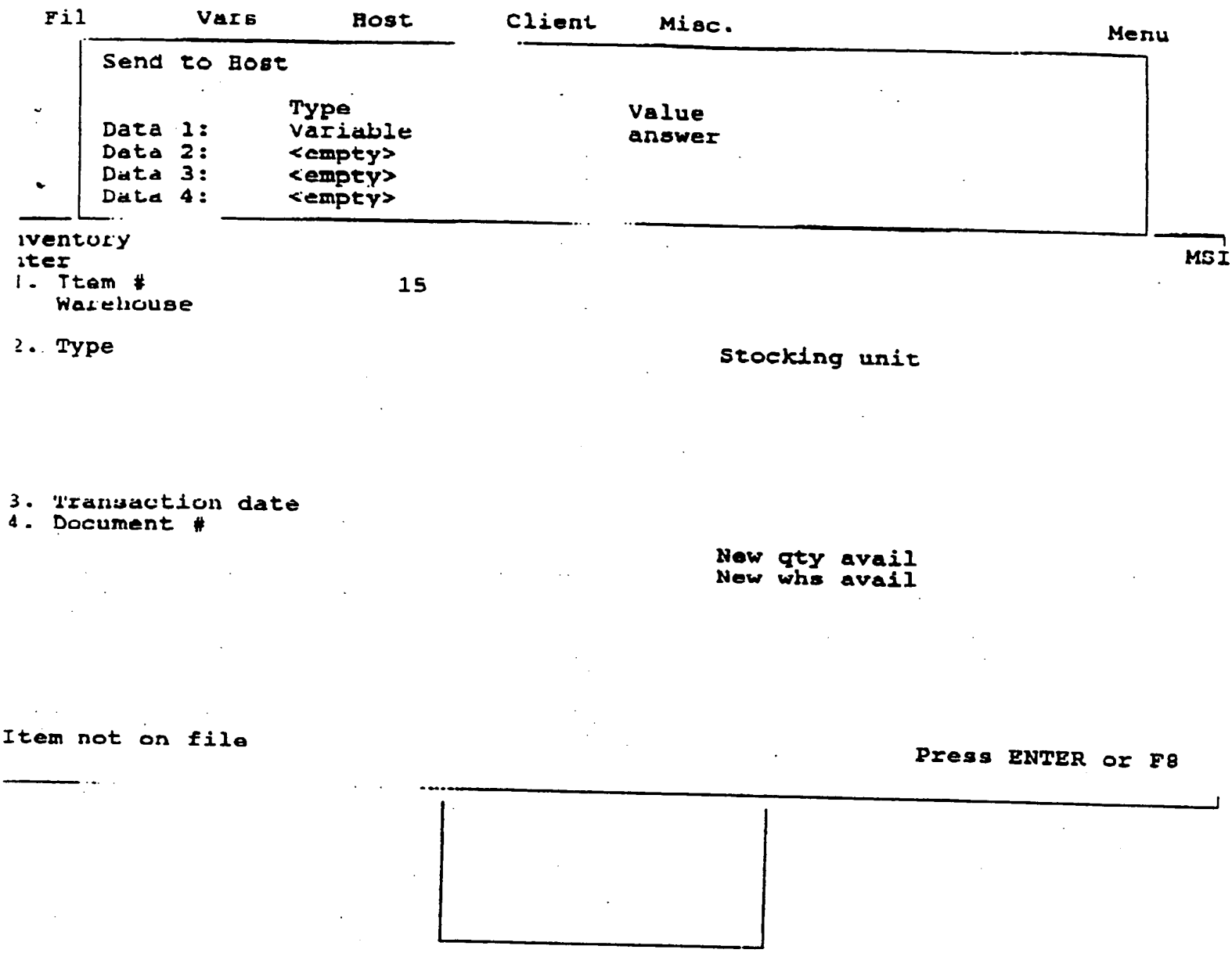

File
Vars
Host
Client
Misc.
Monitor
HOST
ventory
ter
. It m #
Warehouse
15
MSI
. Type
Stocking unit
. Transaction date
. Docum nt #
New qty avail
whs avail
CLIENT
Item not on file
Press ENTER to
continue:
tem not on file
Press ENTER or F8

Fig II


Fil
Vars
Host
Client
Misc.
Menu
Send to Host
Type
Value
Data 1:
Variable
answer
Data 2:
<empty>
Data 3:
<empty>
Data 4:
<empty>
ıventory
ıter
MSI
1. Item #
15
Warehouse
2. Type
Stocking unit
3. Transaction date
4. Document #
New qty avail
New whs avail
Item not on file
Press ENTER or F8

10

Fig 11

31

File Vars Host Client Misc. Monitor

HOST

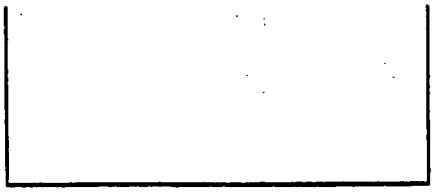
nv ntory MSI
nter
1. Item #
Warehouse

2. Type Stocking unit

3. Transaction date
4. Document #

New qty avail
New whs avail

F1 = next entry F2 = next item blank = look up by description



KK-3

11

FIG KK-a

73

74/75

File Vars Host Cli nt Misc. Menu

Select Link Destination

Path Name: item_transfer
Index: 0

HOST

ventory
ter
. Item #
Warehouse

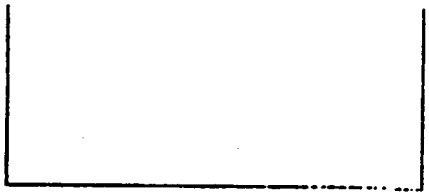
MSI

. Type Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

= next entry F2 = next item blank = look up by description

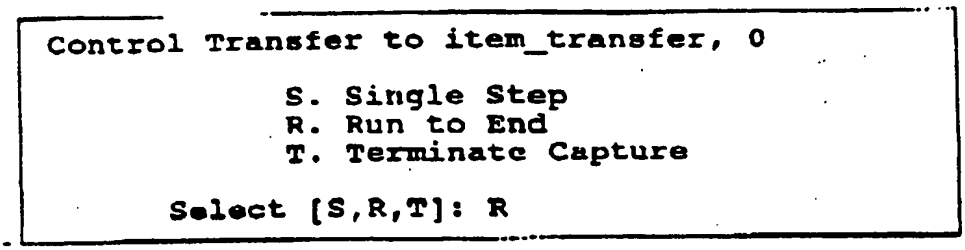


11

VK-C

C11 VKb

33



MSI

ntory
r
. Item #
Warehouse

. Typ

Stocking unit

. Transaction date
. Document #

New qty avail
New whs avail

= n xt entry F2 = next item blank = look up by description

11

FIG KK-C

LV.C

INTERNATIONAL SEARCH REPORT

Inte. al Application No
PCT/US 95/05009

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/455

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	IBM TECHNICAL DISCLOSURE BULLETIN, vol.33, no.5, October 1990, NEW YORK, US page 90 'ADDITION OF NEW EHLLAPI PARAMETER TO HIDE NON-DISPLAY-TYPE FIELDS' see the whole document ---	1
X	IBM TECHNICAL DISCLOSURE BULLETIN, vol.32, no.4A, September 1989, NEW YORK, US pages 290 - 291 'SYSTEM FOR ACCESSING A MAINFRAME FROM A WORKSTATION USER INTERFACE' see the whole document --- -/--	1

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 June 1995	18.07.95
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Authorized officer Fonderson, A

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Intern... al Application No
PCT/US 95/05009

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	IBM TECHNICAL DISCLOSURE BULLETIN, vol.33, no.3B, August 1990, NEW YORK, US page 132 'INTELLIGENT KEYSTROKE CAPTURE FOR PERSONAL COMPUTERS' see the whole document ---	1
A	DR DOBB'S JOURNAL, vol.16, no.3, March 1991, US pages 70 - 71, 148 - 149 DAN TROY: 'Remote Connectivity for Portable Terminals Part II' see the whole document -----	1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)